AD/A-002 976

COMPUTER LISTINGS FOR ILLIAC IV VERSION OF FKCOMB

Ann Kerr, et al

Teledyne Geotech

Prepared for:

Air Force Technical Applications Center
Defense Advanced Research Projects Agency

6 November 1974

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER SDAC-TR-74-18	2 GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER AD/A-ØØ2976
4 TITLE (and Subtitle) COMPUTER LISTINGS FOR ILLIAC IV VERSIONS OF FKCOMB		5 TYPE OF REPORT & PERIOD COVERED Technical
		6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s) Kerr, Ann and Wagenbreth, Gene		8 CONTRACT OR GRANT NUMBER(s) F08606-74-C-0006
9 PERFORMING ORGANIZATION NAME AND ADDRESS Teledyne Geotech 314 Montgomery Street Alexandria, Virginia 22314		10 PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Nuclear Monitoring Research Office 1400 Wilson Blvd.-Arlington, Virginia 22209		12 REPORT DATE 6 November 1974
		13 NUMBER OF PAGES 76
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) VELA Seismological Center 312 Montgomery Street Alexandria, Virginia 22314		15 SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

ILLIAC IV
DEM1
DEM2
FKCOMB

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

COMPUTER LISTINGS FOR ILLIAC IV VERSIONS OF FKCOMB

SEISMIC DATA ANALYSIS CENTER REPORT NO.: SDAC-TR-74-18

AFTAC Project No.:	VELA VT/4709
Project Title:	Seismic Data Analysis Center
ARPA Order No.:	1620
ARPA Program Code No.:	3F10
Name of Contractor:	TELEDYNE GEOTECH
Contract No.:	F08606-74-C-0006
Date of Contract:	01 July 1974
Amount of Contract:	$2,237,956
Contract Expiration Date:	30 June 1975
Project Manager:	Royal A. Hartenberger (703) 836-3882

P. O. Box 334, Alexandria, Virginia 22314

TABLE OF CONTENTS

3<

SUMMARY

This is the third report in a series of three published at the Seismic Data Analysis Center in 1974, which describe our studies of and programming experience with the ILLIAC IV computer. The present report is a computer listing of the ILLIAC IV version of a scientific program called FKCOMB. The main program, FKCOMB, and two data-editing and formatting modules, DEM1 and DEM2 were written in Computational Fluid Dynamics Code (CFD); some subroutines were written in ASK code.

The first report in the series, SDAC-TR-74-16, presents an overview of the ILLIAC IV System, describes the suitability of the ILLIAC IV computer as a processor of seismic data, and contains project notes on programming techniques and languages. The second report, SDAC-TR-74-17, is a complete documentation of the preliminary version of the FKCOMB software designed for processing long-period seismic data; it also compares results from the ILLIAC software with those of the original version of FKCOMB.

DEM1

```
C     MAIN DRIVER FOR MOD 1. ENTERED BY GENE WAGENBRETH, APRIL 24, 1974
     *PE INTEGER CNTRL(*,6),OUTBUF(*,64,6),PINT1(*),INBUF(*,128),        -
     1          TIME(*),OLDTIM(*),                                       -
     1                     SAVBCT,SAVPTW,OUPAGE(6),TSTEPS(6),SCANS,      -
     2            OUPTWA(6),      OTIMEA(6)
     *CU INTEGER ADBBUF(8),ARRAY,INPTB,INPTW,SAVADB,ADBOUT(6),OUPTW,     -
     1            BYTS,WORDS,T1,T2,T3,T4,T5,T6,       IT,PRTIAL,ADDRS,   -
     2            WORD,      BYTCNT(6),ADBWRD,INBYT,OUBYT,ORGCOR,PAGE,   -
     3            DEBUG,BCT,ADB,ENDADB
     *CU LOGICAL LADBBU(8),LARRAY,LINPTB,LINPTW,LSAVAD,LADBOU(6),LOUPTW,-
     1            LBYTS,LWORDS,LT1,LT2,LT3,LT4,LT5,LT6,LOUBYT,LIT,LPRTIA,-
     2            LADDRS,LWORD,LINBYT,LBYTCN(6),LADBWR,LORGCO,LPAGE      -
     3            ,LDEBUG,LBCT,LADB,LENDAD
     *EXTERNAL RDPRM,GETBYT,PUTBYT,CNVTIM
     *COMMON/MAIN/CNTRL,OUTBUF,INBUF,PINT1,TIME,OLDTIM,SAVBCT,SAVPTW,    -
     1      TSTEPS,SCANS,OUPTWA,OUPAGE,       OTIMEA
     *EQUIVALENCE(1,ADBBUF(1),LADBBU(1)),(9,ARRAY,LARRAY),              -
     1            (10,INPTB,LINPTB),                                     -
     1            (11,INPTW,LINPTW),(12,SAVADB,LSAVAD),                  -
     1            (13,ADBOUT(1),LADBOU(1))                               -
     2            ,(19,OUPTW ,LOUPTW),(20,BYTS,LBYTS),(21,WORDS,LWORDS), -
     3            (22,T1,LT1),(23,T2,LT2),(24,T3,LT3),(25,T4,LT4),       -
     4            (26,T5,LT5),(27,T6,LT6),(28,OUBYT,LOUBYT),(29,IT,LIT), -
     5            (30,PRTIAL,LPRTIA),(31,ADDRS,LADDRS),(32,WORD,LWORD),  -
     6            (33,INBYT,LINBYT),(34,BYTCNT(1),LBYTCN(1)),            -
     6            (40,ADBWRD,LADBWR)                                     -
     7                                                ,(43,ORGCOR,LORGCO),-
     8            (44,PAGE,LPAGE),(45,DEBUG,LDEBUG),(46,BCT,LBCT),       -
     9            (47,ADB,LADB),(48,ENDADB,LENDAD)
     *DISK AREA OUPUT1(20),OUPUT2(20),OUPUT3(20),OUPUT4(20),OUPUT5(20), -
     1          OUPUT6(20),INPUT(100)
      MODE=ON
      ENDADB=-1
      ORGCOR=-8193
A     JUMP AROUNDAREA;
A     DISPA:AREA "DISPA";
A     AROUNDAREA::OPNDISP DISPA;
C     THAT IS USED FOR DISPLAY OUTPUT.
A     DISPLS " ", 16,BEGINHEADER,ENDHEADER-1;
A     JUMP ENDHEADER;
A     BLK;
A     BEGINHEADER:::
A     DATA (("********")8, 3338)2,
A     "DATA EDITING MODULE 1 VERSION 1.1",000A:16,
A     (("********")8,3338)2;
A     ENDHEADER:::
     *CALL RDPRM
C     SUBROUTINE RDPRM INITIALIZES VALUES TO BE USED BY THIS AND
C     FOLLOWING MODULES.
  10 *CALL GETBYT
```

```
C       THE HEADER ID HAS  JUST BEEN READ IN. NOW TO LOOK IT UP.
      *DO 20 ARRAY=1,7
      *IF (ARRAY.EQ.7) GO TO 850
C       IF WE GET THAT FAR, THE HEADER ID WAS NOT ON OUR LIST.
       T6=CNTRL(11,ARRAY)
      *IF(T6.EQ.INBYT)GO TO 25
C       THAT MEANT WE FOUND IT.
   20 *CONTINUE
   25 *CONTINUE
      *IF (DEBUG.LT.1) GO TO 35
A       DISPLS " ", 16,B1,E1-1;
A       SKIP,E1;
A       B1:DATA "GOT THE HEADER ID SUCCESSFULLY.",ODOA:16;
A       E1:DISPLH "ARRAY:",2;
   35 *CONTINUE
   40  T6=CNTRL(1,ARRAY)
      *IF (T6.EQ.0)GO TO 45
C       THAT CHECKED TO SEE IF THERE IS TIMING INFO TO GET FROM THE
C       BEGINNING OF THIS RECORD.
      *CALL CNVTIM
C       CNVTIM READS OFF THE TIME AND CONVERTS IT TO DECISECONDS FROM
C       THE BEGINNING OF THE YEAR.
   45 *CONTINUE
       OLDTIM(*)=OTIMEA(ARRAY)
C       HALF TO RESTORE THE VALUE LEFT IN OLDTIM WHEN THE LAST RECORD
C       FROM THIS ARRAY WAS PROCESSED.
       T6=CNTRL(4,ARRAY)
       INPTB=INPTB+T6
C       THAT MOVED THE INPUT POINTER TO THE BEGINNING OF A TIME SCAN.
C       START PROCESSING A TIME SCAN.
       SCANS=0
   50 *IF (DEBUG.LT.1) GO TO 55
A       DISPLH "T-SCAN",0;
   55  T6=CNTRL(5,ARRAY)
       INPTB=INPTB+T6
       T6=CNTRL(1,ARRAY)
      *IF(T6.EQ.1)GO TO 65
C       IF WE GET HERE THERE IS A TIME WORD WITH THIS SCA.
      *CALL CNVTIM
       T6=CNTRL(6,ARRAY)
       INPTB=INPTB+T6
   65 *IF (DEBUG.LT.1) GO TO 75
A       DISPLH "TIME",16,TIME,TIME;
A       DISPLH "OLDTIM",16,OLDTIM,OLDTIM;
   75 *CONTINUE
C       NOW CHECK TO SEE IF A TIME SCAN IS MISSING.
      *IF(.ANY.((OLDTIM(*).EQ.0)))GO TO 200
      *IF(.ANY.((TIME(*).LE.OLDTIM(*)+15)))GO TO 200
      *IF(.ANY.((TIME(*).GT.OLDTIM(*)+25)))GO TO 150
C       IF WE GET HERE THERE WERE 1 OR 2 TIME SCANS MISSING. WE HAVE TO
C       FILL IN THE MISSING TIME GAP(S) WITH THE LAST TIME GAP, WHICH HAS
```

```
C     BEEN CAREFULLY SAVED IN OUTBUF. WE DO SO A BYTE AT A TIME UNTIL
C     ALL CHANNELS ARE DONE. THIS IS NOT THE FASTEST WAY TO DO THE JOB
C     BUT IS STRAIGHT FORWARD AND EASILY DEBUGGABLE. THE SITUATION IS
C     COMPLICATED A BIT BY THE FACT THAT SOME OF THE BYTES ARE IN
C     ADBOUT(ARRAY).
C
      SAVADB=ADBOUT(ARRAY)
      SAVBCT=BYTCNT(ARRAY)
      T6=OUPTW-1
      SAVPTW=T6
      T6=CNTRL(3,ARRAY)
      T5=SAVBCT
      BYTS=T6-T5
C     NUMBER OF BYTES TO TRANSFER FROM OUTBUF IS EQUAL TO THE NUMBER OF
C     CHANNELS MINUS THE NUMBER OF BYTES IN ADB.
A     % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      LWORDS=LBYTS.SHR.2
      LT1=LWORDS.SHL.2
A     %     DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      PRTIAL=BYTS-T1
      T6=SAVPTW
      ADDRS=T6-WORDS
C     ADDRESS IN OUTBUF OF PARTIAL WORD.
C     NOW I WOULD LIKE TO HAVE OUTBUF DIMENSIONED "OUTBUF(8192,8)" AND
C     SIMPLY ACCESS "OUTBUF(ADDRS,ARRAY)" BUT CFD FORCES ME TO
C     DIMENSION OUTBUF "OUTBUF(*,64,8)" AND I HAVE TO DO SOME ARITHMETIC
C     HERE TO CALCULATE THE PROPPER INDICES.
      T1=ADDRS
A     %     DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      LT2=OFF.TURN ON..LAST.6
      LT2=LT2.AND.LT1
      LT1=LT1.SHL.6
A     %     DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      IT=OUTBUF(T2,T1,ARRAY)
     *IF (DEBUG.LT.1) GO TO 105
A     DISPLH "TIME-GAP",0;
A     DISPLH "SAVADB",16,SAVADB,SAVADB;
A     DISPLH "SAVBCT",16,SAVBCT,SAVBCT;
A     DISPLH "SAVPTW",16,SAVPTW,SAVPTW;
A     DISPLH "BYTS",16,BYTS,BYTS;
A     DISPLH "WORDS",16,WORDS,WORDS;
A     DISPLH "PRTIAL",16,PRTIAL,PRTIAL;
A     DISPLH "ADDRS",16,ADDRS,ADDRS;
A     DISPLH "IT",16,IT,IT;
 105 *CONTINUE
C     NOW WE COME TO ANOTHER CFD ABOMINATION. I WANT TO DIVIDE ONE
C     NUMBER IN MEMORY BY 4. I CAN EITHER MOVE IT TO A VECTOR OR
C     FIDDLE WITH THE MODE OR SOMETHING AND DO A STRAIGHT DIVIDE,
C     OR I CAN MOVE IT TO THE CU AND JUST SHIFT IT. I OPT FOR THE LATTER
C     AND CONTINUE TO OPT FOR THAT THRUOUT THE PROGRAM. THIS IS BECAUSE
C     SOMEDAY CFD MAY ALLOW ME TO DO WHAT I WANT OR I CAN PUT IT IN CODE
```

```
C     STATEMENTS MYSELF.
A      %     DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LPRTIA=LPRTIA.SHL.2
       LIT=LIT.RTR.PRTIAL
A      %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.

       PINT1(*)=TIME(*)-OLDTIM(*)
       T6=PINT1(1)
      *DO 140 T1=1,T6,10
C      ONCE FOR EACH MISSING TIME GAP
C      FIRST DO THE PARTIAL WORD.
       TIME(*)=TIME(*)+10
C      LOUBYT=TIME
A      SLIT(0) =TIME:
A      LOAD(0) $C0:
A      CSHR(0) 16:
A      STL(0) LOUBYT:
      *CALL PUTBYT
A      SLIT(0) TIME:
A      LOAD(0) $C0:
A      LIT(1)=0FFFF:16:
A      CAND(0) $C1:
A      STL(0) LOUBYT:
      *CALL PUTBYT
      *DO 110 T2=1,PRTIAL,16
C      ONCE FOR EACH BYTE.
       LIT=LIT .RTL.16
       LOUBYT=OFF.TURN ON..LAST.16
       LOUBYT=LOUBYT.AND.LIT
 110  *CALL PUTBYT
C      NOW DO THE FULL WORDS.
       T6=SAVPTW
      *DO 120 T2=ADDRS+1,T6
C      NOW WE HAVE TO DO THE MESSED UP ADDRESSING BECUASE OF THE CFD
C      RESTRICTION ON DIMENSIONING.
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT3=LT2.SHR.6
       LT4=OFF.TURN ON..LAST.6
       LT4=LT4.AND.LT2
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       WORD=OUTBUF(T4,T3,ARRAY)
      *DO 120 T5=1,4
       LWORD=LWORD.RTL.16
       LOUBYT=OFF.TURN ON..LAST.16
       LOUBYT=LOUBYT.AND.LWORD
 120  *CALL PUTBYT
C      NOW PUT OUT THE BYTES IN SAVADR.
       WORD=SAVADR
       T6=SAVBCT
      *DO 140 T2=1,T6
       LOUBYT=OFF.TURN ON..LAST.16
       LOUBYT=LOUBYT.AND.LWORD
      *CALL PUTBYT
 140  LWORD=LWORD.RTL.16
     *GO TO 250
 150 *CONTINUE
C      IF WE GET HERE, THERE WWERE A BUNCH OF TIME GAPS MISSING. WE NEED
C      ONLY PUT OUT A DIAGNOSTIC
A      DISPLS " ", 16,BGAPDIAGNOSTIC,EGAPDIAGNOSTIC-1;
A      DISPLH "OLDTIM",16,OLDTIM,OLDTIM;
A      DISPLH "TIME",16,TIME,TIME;
A      JUMP EGAPDIAGNOSTIC;
A      BLK;
A      BGAPDIAGNOSTIC:::
A      DATA (("*********")8,3338)10,"MORE THAN 2 TIME GAPS.",ODOA:16,
A           (("*********")8,3338)10;
A      EGAPDIAGNOSTIC:::
 175 *CONTINUE
 200 *CONTINUE
C      NOW TO CHECK FOR A TIME REVERSAL.
     *IF(.ANY.((TIME(*).GT.OLDTIM(*)+5)))GO TO 250
C      IF WE GET HERE THERE IS A TIME REVERSAL. IF THERE IS A TIME WORD
C      PER SCAN, THIS SCAN IS THROWN OUT. OTHERWISE, THE WHOLE RECORD
C      IS DISCARDED.
       T6=CNTRL(1,ARRAY)
     *IF (T6.EQ.1)GO TO 210
C      WE HAVE A TIME WORD PER SCAN.
       PINT1(*)=INPTB+CNTRL(6,ARRAY)+CNTRL(7,ARRAY)+CNTRL(3,ARRAY)
       INPTB=PINT1(1)
C      NOW SKIP AROUND PROCESSING OF THIS SCAN.
     *GO TO 410
C      NOW HANDLE THROWING AWAY THE WHOLE RECORD FOR TIME WORD PER
C      RECORD.
 210   PINT1(*)=CNTRL(4,ARRAY)+CNTRL(9,ARRAY)*(CNTRL(5,ARRAY)+
      1CNTRL(3,ARRAY)+CNTRL(7,ARRAY))+CNTRL(8,ARRAY)+INPTB
       INPTB=PINT1(1)
       T6=SCANS
       T6=T6+100
       SCANS=T6
     *GO TO 410
C      FINALLY TIME IS ALL TAKEN CARE OF. FIRST WE OUTPUT TIME. THEN ALL
C      THE DATA.
 250 *CONTINUE
C      LOUBYT=TIME(1)
A      SLIT(O) TIME;
A      LOAD(O) $CO;
A      CSHR(O) 16;
A      STL(O) LOUBYT;
     *CALL PUTBYT
A      SLIT(O) TIME;
A      LOAD(O) $CO;
A      LIT(1) =OFFFF:16;
A      CAND(O) $C1;
```

```
A      STL(O) LOUBYT:
      *CALL PUTBYT
       OLDTIM(*)=TIME(*)
       TIME(*)=TIME(*)+10
       T6=TSTEPS(ARRAY)
       T6=T6+1
       TSTEPS(ARRAY)=T6
       T6=CNTRL(3,ARRAY)
      *DO 400 T1=1,T6
C      THATS ONCE FOR EACH CHANNEL.
      *CALL GETBYT
       OUBYT=INBYT
      *CALL PUTBYT
 400  *CONTINUE
 410   T6=SCANS
       T6=T6+1
       SCANS=T6
       T6=CNTRL(7,ARRAY)
       INPTB=INPTB+T6
       T6=CNTRL(9,ARRAY)
       T5=SCANS
      *IF(T5.GE.T6)GO TO 420
      *GO TO 50
 420   T6=CNTRL(8,ARRAY)
       INPTB=INPTB+T6
       OTIMEA(ARRAY)=OLDTIM(1)
      *GO TO 10
 850  *CONTINUE
      *IF(INBYT.EQ.0)GO TO 855
A      DISPLH "BADHEAD:",2:
 855  *CONTINUE
C      WHEN WE GET HERE ALL THE DATA HAS BEEN PROCESSED. TIME TO EMPTY
C      THE ADB BUFFERS, WRITE OUT THE CORE BUFFERS , PUT THE HEADERS IN
C      THE OUTPUT FILES AND LAY BACK AND QUIT.
C
C      FIRST EMPTY THE ADB BUFFERS.
      *DO 860 ARRAY=1,6
       BCT=4-BYTCNT(ARRAY)
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LBCT=LBCT.SHL.4
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LADB=LADBOU(ARRAY).SHL.BCT
C      HERE WE HAVE TO DO SOME MORE FANCY ADDRESSING BECAUSE OF THE CFD
C      RESTRICTION ON DIMENSIONS.
       T1=OUPTW
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT2=LT1.SHR.6
       LT6=OFF.TURN ON..LAST.6
       LT1=LT6.AND.LT1
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       OUTBUF(T1+1,T2+1,ARRAY)=ADB
 860  *CONTINUE
```

```
C     NOW WRITE OUT ALL OF THE BUFFERS(6 AT THIS TIME).
      T6=OUPAGE(1)
     *WRITE(64,OUTBUF(1,1,1),OUPUT1(T6),4)
     *WAIT 64
      T6=OUPAGE(2)
     *WRITE(64,OUTBUF(1,1,2),OUPUT2(T6),4)
     *WAIT 64
      T6=OUPAGE(3)
     *WRITE(64,OUTBUF(1,1,3),OUPUT3(T6),4)
     *WAIT 64
      T6=OUPAGE(4)
     *WRITE(64,OUTBUF(1,1,4),OUPUT4(T6),4)
     *WAIT 64
      T6=OUPAGE(5)
     *WRITE(64,OUTBUF(1,1,5),OUPUT5(T6),4)
     *WAIT 64
      T6=OUPAGE(6)
     *WRITE(64,OUTBUF(1,1,1),OUPUT6(T6),4)
     *WAIT 64
C     NOW FOR THE HEADER PAGE. FIRST CLEAR A BUFFER.
      *DO 930 T1=1,16
 930  OUTBUF(*,T1,1)=0
C     NOW FILL THEM IN ONE AT A TIME.
      OUTBUF(1,1,1)=CNTRL(11,1)
C     THAT WAS THE HEADER ID.
      OUTBUF(2,1,1)=TSTEPS(1)
     *WRITE(64,OUTBUF(1,1,1),OUPUT1(1),1)
     *WAIT 64
      OUTBUF(1,1,1)=CNTRL(11,2)
      OUTBUF(2,1,1)=TSTEPS(2)
     *WRITE(64,OUTBUF(1,1,1),OUPUT2(1),1)
     *WAIT 64
      OUTBUF(1,1,1)=CNTRL(11,3)
      OUTBUF(2,1,1)=TSTEPS(3)
     *WRITE(64,OUTBUF(1,1,1),OUPUT3(1),1)
     *WAIT 64
      OUTBUF(1,1,1)=CNTRL(11,4)
      OUTBUF(2,1,1)=TSTEPS(4)
     *WRITE(64,OUTBUF(1,1,1),OUPUT4(1),1)
     *WAIT 64
      OUTBUF(1,1,1)=CNTRL(11,5)
      OUTBUF(2,1,1)=TSTEPS(5)
     *WRITE(64,OUTBUF(1,1,1),OUPUT5(1),1)
     *WAIT 64
      OUTBUF(1,1,1)=CNTRL(11,6)
      OUTBUF(2,1,1)=TSTEPS(6)
     *WRITE(64,OUTBUF(1,1,1),OUPUT6(1),1)
     *WAIT 64
A     DISPLS " ", 16,BFINAL,EFINAL-1;
A     JUMP EFINAL;
A      BLK;
```

```
A     BFINAL::
A     DATA (("*********")8,3338)10,
A     "MOD 1 GOING TO END OF JOB",CDOA:16,(("*********")8,3338)10;
A     EFINAL::;
A
A     CLSDISP DISPA;
     *STOP
     *END
```

DEM2

```
C                         DEM2 - DATA EDITTING MODULE 2
C        CODED BY GENE WAGENBRETH MAY, 1974. THIS ROUTINE READS IN DATA
C        FROM ONE SEISMIC ARRAY AND PERFORMS ALL DATA EDITTING UP TO AND
C        INCLUDING FFT.
C
C
      *PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),     -
      1           PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
      *PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
      1        PREAL2(*),ALLMSQ(*),TVARFT(*)
      *PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
      1           CHGOOD(80),SITEGD(80),SITES(80)
      *PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
      *CU INTEGER ADBBUF(8),COREPT,      BYTE,ADBWRD,ARRAY,DEBUG,TWSZ,   -
      1           OVLAP,NCHAN,NSITE,NROWS,DIFFR,DIFFW,NEW,OLD,GAP,TSCANS,-
      2          INDEX1,INDEX2,INDEX3,INDEX4,T1,T2,T3,T4,T5,T6,CH,IPAGE, -
      3          OFFSET,INBYT,NGDCH,TWSZR,NGDST,NGDR,F,BF3PE,NGT,OPAGE,T7
      *CU LOGICAL LADBBU(8),LCOREP,LAST16,LBYTE,LADBWR,LARRAY,LDEBUG,     -
      1           LTWSZ,LOVLAP,LNCHAN,LNSITE,LNROWS,LDIFFR,LDIFFW,LNEW,  -
      2           LOLD,LGAP,LTSCAN ,LT1,LT2,LT3,LT4,LT5,LT6,LCH,LOFFSE,  -
      3           LINBYT,LF,LNGDCH,LTWSZR,LNGDST,LNGDR,LNGT,LT7
      *EXTERNAL GTDATA,C16T64,C64T32,ROWSUM,RUNFFT,C32T64
      *COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,   -
      1        TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,   -
      2        ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
      *EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),      -
      1             ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),          -
      2           (BUFF3(1,1),IBUFF3(1,1))
      *EQUIVALENCE (1,ADBBUF(1),LADBBU(1)),(9,COREPT,LCOREP),(10,BYTE,   -
      1           LBYTE),(11,ADBWRD,LADBWR),(12,ARRAY,LARRAY),(13,TWSZ,  -
      2           LTWSZ),(14,OVLAP,LOVLAP),(15,NCHAN,LNCHAN),(16,NSITE,  -
      3           LNSITE),(17,NROWS,LNROWS),(18,DIFFR,LDIFFR),(19,DIFFW, -
      4           LDIFFW),(20,NEW,LNEW),(21,OLD,LOLD),(22,GAP,LGAP),(23, -
      5           TSCANS,LTSCAN),(24,INDEX1),(25,INDEX2),(26,INDEX3),(27,-
      6           INDEX4),(28,T1,LT1),(29,T2,LT2),(30,T3,LT3),(31,T4,    -
      7           LT4),(32,T5,LT5),(33,T6,LT6),(34,CH,LCH),(35,OFFSET,   -
      8           LOFFSE),(36,INBYT,LINBYT),(37,F,LF),(38,NGDCH,LNGDCH), -
      9           (39,TWSZR,LTWSZR),(40,NGDST,LNGDST),(41,NGDR,LNGDR),   -
      0           (42,BF3PE),(43,NGT,LNGT),(44,LAST16),(45,DEBUG,LDEBUG),-
      1           (46,OPAGE),(47,T7,LT7),(48,IPAGE)
      *DISK AREA INDM2(20),OUTDM2(41),CONPRM(1)
       MODE=ON
A      JUMP PASTAREA;
A      DISP2::AREA "DISP2";
A      PASTAREA:::
A      OPNDISP DISP2;
A      DISPLS ,16,BHEAD2,EHEAD2-1;
A      JUMP EHEAD2;
A      BLK;
A      BHEAD2::DATA
A      (("********")8,ODOA:16)10,"START EXECUTION DATA EDITING MODULE 2",
A      ODOA:16,"VERSION 2.0",ODOA:16,(("********")8,ODOA:16)2;
A      EHEAD2::;
      *READ(64,IBUFF1(1),INDM2(1),1)
      *WAIT 64
      *WRITE(64,IBUFF1(1),OUTDM2(1),1)
      *WAIT 64
       T1=IBUFF1(1)
      *DO 20 ARRAY=1,7
      *IF(ARRAY.EQ.7)GO TO 1105
       T2=CNTRL(1,ARRAY)
      *IF(T1.EQ.T2)GO TO 25
C      THAT MEANT WE FOUND IT.
   20 *CONTINUE
   25 *CONTINUE
       FINSCN(*)=IBUFF1(2)
      *READ(64,IBUFF1(1),CONPRM(1),1)
      *WAIT 64
       DEBUG=IBUFF1(1)
       TWSZ=IBUFF1(2)
       OVLAP=IBUFF1(3)
       GLCHFT(*)=RBUFF1(4)
       VARFT(*)=RBUFF1(5)
       LOFREQ=IBUFF1(6)
       HIFREQ=IBUFF1(7)
       COMP(*)=IBUFF1(8)
       NCHAN=CNTRL(3,ARRAY)
       NSITE=CNTRL(4,ARRAY)
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LTWSZR=LTWSZ.SHR.6
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       PINT1(*)=NCHAN*TWSZR
       NROWS=PINT1(1)
       DIFFW=TWSZ-OVLAP
A      % DUMMY ASK STEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LDIFFR=LDIFFW.SHR.6
       LT1=OFF.TURN ON..LAST.6
       LDIFFW=LDIFFW.AND.LT1
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LAST16=OFF.TURN ON..LAST.16
       BYTE=4
       ADBWRD=8
       COREPT=4096
       OPAGE=2
       IPAGE=2
       NEW=2
       OLD=1
       GAP=1
      *IF(DEBUG.LT.1)GO TO 30
```

```
A       DISPLH "INIT:",2;
   30 *CONTINUE
      *CALL GTDATA
       TIME(*)=INBYT*65536
      *CALL GTDATA
       TIME(*)=TIME(*)+INBYT
  100 *CONTINUE
C      100 BEGINS THE LOOP GONE THRU ONCE FOR EACH COMPLETE TIME WINDOW.
       TSCANS=0
       TWTIME(*)=TIME(*)
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT1=OFF.TURN ON..LAST.1
       LNEW=LNEW.AND.LT1
       LOLD=LOLD.AND.LT1
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       NEW=NEW+1
       OLD=OLD+1
      *IF(DEBUG.LT.1)GO TO 105
A       DISPLH "TWD",0;
      *IF(DEBUG.LT.2)GO TO 105
A       DISPLH ,2;
  105 *CONTINUE
C      NOW CHECK TO SEE IF WE DETECTED A TIME GAP LAST TIME.
      *IF(GAP.EQ.1)GO TO 200
C      WE CANT OVERLAP IF THERE WAS A GAP.
      *IF(DEBUG.LT.1)GO TO 115
A       DISPLH "OVLAP",0;
  115 *CONTINUE
       TWTIME(*)=TWTIME(*)-OVLAP*10
       PINT1(*)=0
      *IF((PEN(*).GT.64-DIFFW))PINT1(*)=1
C      PINT1(*) INSURES THAT VALUES ROUTED ACROSS A ROW BOUNDARY GO TO
C      THE PRECEDING ROW.
       T1=NROWS-DIFFR
      *IF(T1.EQ.0)GO TO 140
      *DO 130 INDEX1=1,T1
       INDEX2=INDEX1+DIFFR
       BUFF2(*,INDEX1,NEW)=BUFF2(*+DIFFW,INDEX2+PINT1(*),OLD)
C      THAT MOVED ALL THE DATA WANTED FROM THE OLD TIME WINDOW, PLUS
C      SOME GARBAGE. THE GARBAGE IS OKAY BECAUSE IT WILL BE OVERWRITTEN
C      BY GOOD DATA LATER.
  130 *CONTINUE
  140 *CONTINUE
       TSCANS=OVLAP
      *IF(DEBUG.LT.1)GO TO 145
A       DISPLH "E OVLAP",0;
      *IF(DEBUG.LT.2)GO TO 145
A       DISPLH "BUFF21",18,BUFF2,BUFF2+255;
A       DISPLH "BUFF22",16,BUFF2+550*64,BUFF2+550*64+255;
  145 *CONTINUE
  200 *CONTINUE
```

```
C      NOW ITS TIME TO READ IN AND MOVE A TIME STEP.
      *IF(DEBUG.LT.1)GO TO 205
A      DISPLH "TSTEP",0;
 205  *CONTINUE
       CH=1
      *DO 280 T1=TSCANS+1,TWSZ
C      THATS ONCE FOR EACH TIMESTEP TO GET FOR THIS TIME WINDOW.
       TOTSCN(*)=TOTSCN(*)+1
       OFFSET=0
      *IF(NEW.EQ.2)OFFSET=35200
C      THAT MAKES ADDRESSING A LITTLE BIT EASIER. WE CAN TREAT BUFF2
C      (ABUFF2) AS 1 DIMENSIONAL. OFFSET ACTS LIKE (NEW,OLD). ITS REALLY
C      BECAUSE CFD WILL NOT ALLOW A 2 DIMENSIONAL ARRAY WITH THE FIRST
C      DIMENSION OTHER THAN 64.
      *DO 240 T2=1,NCHAN
      *CALL GTDATA
       INDEX1=CH+OFFSET
       ABUFF2(INDEX1)=INBYT
       OFFSET=OFFSET+TWSZ
 240  *CONTINUE
C      WE JUST DID ONE TIME STEP.
       TSCANS=TSCANS+1
       CH=CH+1
      *IF(DEBUG.LT.2)GO TO 245
A      DISPLH "1-CHAN",0;
 245  *CONTINUE
C      NOW WE CHECK FOR A GAP.
       T6=TOTSCN(1)
      *IF(.ANY.((T6.EQ.FINSCN(*))))GO TO 300
C      EOF IS REALLY JUST AN INFINITE GAP.
C      GET THE NEXT TIME WORD AND CHECK FOR A GAP.
      *CALL GTDATA
       T6=TIME(1)
       OTIME(*)=TIME(*)
       TIME(*)=INBYT
      *CALL GTDATA
       TIME(*)=TIME(*)*65536+INBYT
      *IF(DEBUG.LT.2)GO TO 255
A      DISPLH "TIME",16,TIME,TIME;
 255  *CONTINUE
      *IF(.ANY.((TIME(*)-T6.GT.15)))GO TO  300
 280  *CONTINUE
       GAP=0
      *GO TO 400
 300  *CONTINUE
C      IF WE GET HERE WE HAVE A GAP.
      *IF(DEBUG.LT.1)GO TO 315
A      DISPLH "GAP",0;
      *IF(DEBUG.LT.2)GO TO 315
A      DISPLH ,2;
 315  *CONTINUE
      *IF(TSCANS.NE.TWSZ)GO TO 325
```

```
C      IF WE GET HERE, THERE IS A GAP, BUT WE DONT NEED ANYMORE DATA
C      SINCE THE TIME WINDOW IS ALREADY FULL. JUST MARK THE FACT THAT WE
C      HAD A GAP.
       GAP=1
      *GO TO 400
 325  *CONTINUE
C      IF WE GET HERE , WE NEED TO FILL IN SOME DATA FROM THE LAST TIME
C      WINDOW TO COMPLETE THIS TIME WINDOW. FIRST CHECK TO SEE IF THERE
C      WAS A GAP AT THE END OF THE LAST TIME WINDOW, IN WHICH CASE WE
C      HAVE AN IRRECOVERABLE ERROR.
      *IF(GAP.NE.1)GO TO 335
C      WHOOPS, AN IRRECOVERABLE ERROR.
A      JUMP EGAPMESSAGE;
A      BLK;
A      BGAPMESSAGE::DATA
A      (("********")8,ODOA:16)10,
A      "IRRECOVERABLE TIME GAP.",ODOA:16,
A      (("********")8,ODOA:16)10;
A      EGAPMESSAGE::;
A      DISPLS ,16,BGAPMESSAGE,EGAPMESSAGE-1;
A      DISPLH "TIME:",16,TIME,TIME;
A      DISPLH "OTIME:",16,OTIME,OTIME;
A      DISPLH "TWTIME:",16,TWTIME,TWTIME;
      *IF(DEBUG.LT.1)GO TO 330
A      DISPLH ,2;
 330  *CONTINUE
      *GO TO 1000
 335  *CONTINUE
C      IF WE GET HERE ITS TIME TO ACTUALLY FILL IN A TIME GAP. FIRST THE
C      CURRENT WINDOW HAS TO BE "SHIFTED FORWARD". THE AMOUNT TO MOVE
C      IS THE NUMBER OF MISSING SCANS=TWSZ-TSCANS.
       GAP=1
       T3=TWSZ-TSCANS
       TWTIME(*)=TWTIME(*)-T3*10
A     %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT4=LT3.SHR.6
       LT5=OFF.TURN ON..LAST.6
       LT3=LT3.AND.LT5
C      T3 IS THE NUMBER OF WORDS TO ROUTE.
C      T4 IS THE NUMBER OF ROWS TO ROUTE.
       PINT1(*)=0
      *IF((PEN(*).LE.T3))PINT1(*)=1
C      THATS ONE FOR ALL THE PE'S THAT ARE GONNA SEND DATA ACCROSS A ROW
C      BOUNDARY.
      *DO 350 INDEX1=T4+1,NROWS
       INDEX2=INDEX1-T4
       BUFF2(*,INDEX2,NEW)=BUFF2(*-T3,INDEX1-PINT1(*),NEW)
 350  *CONTINUE
C      NOW COMES THE DIFFICULT PART. WE HAVE TO MOVE DATA FROM THE LAST
C      TIME WINDOW WITHOUT MOVING ANY GARBAGE, SINCE THIS TIME WE WOULD
C      BE OVERWRITING GOOD DATA. THE ROUTE AMOUNT IS TSCANS. THE NUMBER
```

```
C      OF ITEMS TO BE MOVED(BEFORE WE ALWAYS DID THEM ALL) IS
C      TWSZ-TSCANS.
C      T3=ROUTE AMOUNT(ROWS).
C      T4=ROUTE AMOUNT(WORDS)
C      T5=NUMBER TO DO(ROWS).
C      T6=NUMBER TO DO(WORDS).
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT3=LTSCAN.SHR.6
       LT4=OFF.TURN ON..LAST.6
       LT4=LT4.AND.LTSCAN
       T6=TWSZ-TSCANS
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT5=LT6.SHR.6
       LT7=OFF.TURN ON..LAST.6
       LT6=LT6.AND.LT7
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      *IF(DEBUG.LT.1)GO TO 355
A      DISPLH "RT3",2;
 355  *CONTINUE
C      NOW DO IT ONCE FOR EACH CHANNEL, ONCE FOR EACH FULL ROW, AND
C      THEN ONCE FOR EACH PARTIAL ROW.
       PINT1(*)=0
      *IF((PEN(*).GT.64-T4))PINT1(*)=1
C      NOW WE WOULD LIKE TO DO "DO 370 T1=1,NROWS,TWSZR" BUT CFD INSISTS
C      THAT THE INCREMENT BE A CONSTANT, SO WE WILL CONSTRUCT THE
C      EQUIVALENT LOOP.
       T1=1
      *DO 370 T7=1,NCHAN
      *IF(T5.EQ.0)GO TO 365
C      IMPROPPER DO LOOPS MUST BE AVOIDED.
      *DO 360 T2=1,T5
       INDEX1=T2+T1-1
       INDEX2=INDEX1+T3
       BUFF2(*,INDEX1,NEW)=BUFF2(*+T4,INDEX2+PINT1(*),OLD)
 360  *CONTINUE
 365  *CONTINUE
C      NOW TO DO THE LAST ROW OF THE CHANNEL. IT IS ROW T5+1. GET TO
C      PLAY WITH THE MODE THIS TIME.
       INDEX1=T1+T5
       INDEX2=INDEX1+T3
C      ROUTE AMOUNT IS THE SAME.
       MODE=(PEN(*).LE.T6)
       BUFF2(*,INDEX1,NEW)=BUFF2(*+T4,INDEX2+PINT1(*),OLD)
       MODE=ON
       T1=T1+TWSZR
 370  *CONTINUE
C      THATS EVERYTHING. WE NOW HAVE A COMPLETE TIME WINDOW.
      *IF(DEBUG.LT.1)GO TO 375
A      DISPLH "EBACKUP",2;
      *IF(DEBUG.LT.2)GO TO 375
A      DISPLH "BUFF21",16,BUFF2,BUFF2+255;
```

```
A      DISPLH "BUFF22",16,BUFF2+64*550,BUFF2+64*550+255;
 375  *CONTINUE
C      THATS IT. WE ARE BACKED UP.
 400  *CONTINUE
C      NOW WE HAVE A TIME WINDOW IN 16 BIT FORMAT IN BUFF2(-,-,NEW). WE
C      WILL LEAVE IT IN NEW AND CAN OVERWRITE THE DATA IN BUFF2(-,-,OLD)
C      SINCE IT WILL NEVER BE USED AGAIN. WE WILL CONVERT TO 64 BIT
C      FLOATING POINT FORMAT AND MOVE FROM BUFF2(NEW,-,-) TO "OLD".ONCE
C      THE DATA IN "OLD" HAS FOUND ITS WAY TO BUFF3, IT WILL BE OVER
C      WRITTEN NEXT TIME AROUND.
      *DO 405 INDEX1=1,550
       BUFF2(*,INDEX1,OLD)=0.0
 405  *CONTINUE
      *DO 410 INDEX1=1,NROWS
      *CALL C16T64(BUFF2(*,INDEX1,NEW),BUFF2(*,INDEX1,OLD))
 410  *CONTINUE
C      DATA IS NOW IN BUFF2(*,-,OLD). IT WILL NOW BE DEGLITCHED, MEAN
C      SQUARE CALCULATED AND CHECKED AND THEN FFT'ED.
       PINT1(*)=0
       PINT2(*)=0
      *IF((PEN(*).EQ.1))PINT1(*)=1
      *IF((PEN(*).EQ.64))PINT2(*)=1
       ALLMSQ(*)=0.0
       T2=1
C      WE WOULD LIKE TO DO "DO 500 CH=1,NROWS,TWSZR" BUT CFD INSISTS
C      THAT THE INCREMENT BE A CONSTANT, SO WE WILL CONSTRUCT AN
C      EQUIVALENT LOOP.
       CH=1
      *DO 500 T7=1,NCHAN
      *IF(DEBUG.LT.2)GO TO 420
       PINT1(1)=CH
A      DISPLH "CH",16,PINT1,PINT1;
 420  *CONTINUE
      *DO 430 T3=0,TWSZR-1
       INDEX1=CH+T3
       PREAL1(*)=ABS(BUFF2(*,INDEX1,OLD)-BUFF2(*-1,INDEX1-PINT1(*),OLD))
       PREAL2(*)=ABS(BUFF2(*-1,INDEX1-PINT1(*),OLD)-BUFF2(*+1,INDEX1+      -
      1PINT2(*),OLD))
      *IF(DEBUG.LT.2)GO TO 422
A      DISPLF "PREAL1",16,PREAL1,PREAL1+63;
A      DISPLF "PREAL2",16,PREAL2,PREAL2+63;
 422  *CONTINUE
       PREAL2(*)=PREAL2(*)*GLCHFT(*)
      *IF(T3.EQ.0)MODE=MODE.AND.(PEN(*).NE.1)
      *IF(T3.EQ.TWSZR-1)MODE=MODE.AND.(PEN(*).NE.64)
      *IF((PREAL1(*).GT.PREAL2(*)))BUFF2(*,INDEX1,OLD)=                   -
      1(BUFF2(*-1,INDEX1-PINT1(*),OLD)+BUFF2(*+1,INDEX1+PINT2(*),OLD))    -
      2/2.0
      *IF(DEBUG.LT.2)GO TO 430
A      SETC(0) E;
A      DISPLH "MODE",1;
       PREAL1(*)=BUFF2(*,INDEX1,OLD)
```

```
A      DISPLF "BUFF2-",16,PREAL1,PREAL1+63;
 430 *CONTINUE
       MODE=ON
C      NOW REMOVE THE BIAS.
       PREAL1(*)=0.0
      *DO 440 T3=0,TWSZR-1
       INDEX1=T3+CH
       PREAL1(*)=PREAL1(*)+ROWSUM(BUFF2(*,INDEX1,OLD))
 440 *CONTINUE
       PREAL1(*)=PREAL1(*)/FLOAT(TWSZ)
      *DO 450 T3=0,TWSZR-1
       INDEX1=T3+CH
       BUFF2(*,INDEX1,OLD)=BUFF2(*,INDEX1,OLD)-PREAL1(*)
      *IF(DEBUG.LT.2)GO TO 450
A      DISPLF "BIAS",16,PREAL1,PREAL1+63;
       PREAL1(*)=BUFF2(*,INDEX1,OLD)
A      DISPLF "A-BIAS",16,PREAL1,PREAL1+63;
 450 *CONTINUE
C      NOW LETS COMPUTE THE MEAN SQUARE FOR EACH CHANNEL AND FOR THE
C      ENTIRE TIME WINDOW. PREAL1 WILL CONTAIN THE MEAN SQUARE FOR
C      THE PARTICULAR CHANNEL. ALLMSQ WILL CONTAIN THE MEAN SQUARE FOR
C      THE ENTIRE TIME WINDOW. CHMSQ(I) CONTAINS THE MEAN SQUARE FOR
C      CHANNEL NUMBER "I".
       PREAL1(*)=0.0
      *DO 470 T3=0,TWSZR-1
       INDEX1=T3+CH
       PREAL1(*)=PREAL1(*)+ROWSUM(BUFF2(*,INDEX1,OLD)**2)
 470 *CONTINUE
       PREAL1(*)=PREAL1(*)/FLOAT(TWSZ)
       CHMSQ(T2)=PREAL1(1)
       T2=T2+1
       ALLMSQ(*)=ALLMSQ(*)+PREAL1(*)
       CH=CH+TWSZR
 500 *CONTINUE
C      WE NOW HAVE TO COMPARE THE CHANNEL MEAN SQUARES AGAINST THE TOTAL
C      MEAN SQUARE AND VARFT TO SEE WHICH ONES ARE BAD. WE MAKE OUR
C      CRITERIA EASIER AND EASIER UNTIL AT LEAST HALF THE CHANNELS PASS,
C      WITH A LIMIT OF TEN TIMES ON OUR PATIENCE.
       ALLMSQ(*)=ALLMSQ(*)/FLOAT(NCHAN)
      *IF(DEBUG.LT.2)GO TO 502
A      DISPLF "ALLMSQ",16,ALLMSQ,ALLMSQ+63;
A      DISPLF "CHMSQ",16,CHMSQ,CHMSQ+50;
 502 *CONTINUE
       LT3=LNCHAN.SHR.1
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
C      THATS HALF THE NUMBER OF CHANNELS, THE NUMBER THAT NEED TO PASS.
       TVARFT(*)=VARFT(*)
C      PUT VARFT IN THE PE'S SO WE CAN GET TO IT EASILY.
      *DO 550 T1=1,10
      *IF(DEBUG.LT.2)GO TO 504
A      DISPLH "VARLOOP",0;
```

```
 504 *CONTINUE
      NGDCH=0
     *DO 540 CH=1,NCHAN
       CHGOOD(CH)=0
       PREAL1(*)=CHMSQ(CH)
C      PREAL1(*)=PREAL1(*)/ALLMSQ(*)
C     *IF(.ANY.((PREAL1(*).GT.TVARFT(*))))GO TO 510
C     *IF(.ANY.((PREAL1(*).LT.(1.0/TVARFT(*)))))GO TO 510
       CHGOOD(CH)=1
       NGDCH=NGDCH+1
 510 *CONTINUE
 540 *CONTINUE
     *IF(NGDCH.GT.T3)GO TO 560
      TVARFT(*)=TVARFT(*)*1.25
 550 *CONTINUE
 560 *CONTINUE
C     NOW WE HAVE MARKED THE BAD CHANNELS. TIME TO COMPUTE MOTION
C     COMPONENTS AND MARK THE USEFUL COMPONENTS. SINCE ALL THE
C     ARRAYS HAVE THE DATA ARRANGED DIFFERENTLY, WE HAVE A SEPARATE
C     SECTION OF CODE FOR EACH ARRAY.
C      LASA=ARRAY 1; ALPA=ARRAY 2; NORSAR=ARRAY 3
     *IF(ARRAY.NE.1)GO TO 600
C     LASA DATA. CHANNELS ARE ARRANGED VVVV...NNNN...EEEE
C     WE NEED ONLY MARK THE UNINTERESTING CHANNELS.
     *IF(.ANY.((COMP(*).NE.0)))GO TO 580
C     0 MEANS VERTICAL.
     *DO 570 CH=NSITE+1,NCHAN
      CHGOOD(CH)=0
 570 *CONTINUE
     *GO TO 800
 580 *CONTINUE
C     HORIZONTAL MOTION PROCESSED HERE. NOT IMPLEMENTED YET.
A     DISPLH "ARG580",2;
     *GO TO 1100
 600 *CONTINUE
     *IF(ARRAY.NE.2)GO TO 650
C     ALPA DATA CHANNELS. CHANNELS ARE AT 120 DEGREE ANGLES AND SOME
C     COMPUTATION MUST BE DONE. ARRANGED 123123123...
     *IF(.ANY.((COMP(*).NE.0)))GO TO 630
C     0 MEANS VERTICAL.
C     INDEX1=1,2,3,4...NSITE (INC 1)
C     INDEX2=1, (INC TWSZR)
C     INDEX3=1, (INC TWSZR*3) 3 COMPONENTS PER SITE.
C     INDEX4=1, (INC 3)
      INDEX2=1
      INDEX3=1
      INDEX4=1
      T1=TWSZR+TWSZR+TWSZR
     *DO 620 INDEX1=1,NSITE
      T2=0
      T6=CHGOOD(INDEX4)
      *IF(T6.EQ.0)GO TO 610
       T6=CHGOOD(INDEX4+1)
      *IF(T6.EQ.0)GO TO 610
       T6=CHGOOD(INDEX4+2)
      *IF(T6.EQ.0)GO TO 610
       T2=1
C     ALL THREE COMPONENTS MUST BE GOOD FOR A SITE TO BE GOOD.
      *DO 605 T3=0,TWSZR-1
       T4=INDEX2+T3
       T5=INDEX3+T3
       T6=INDEX3+T3+TWSZR
       T7=INDEX3+TWSZR+TWSZR
       BUFF2(*,T4,OLD)=.57735*(BUFF2(*,T5,NEW)+BUFF2(*,T6,NEW)+
      1                  BUFF2(*,T7,NEW))
 605  *CONTINUE
       INDEX2=INDEX2+TWSZR
 610   CHGOOD(INDEX1)=T2
       INDEX3=INDEX3+T1
       INDEX4=INDEX4+3
 620  *CONTINUE
      *GO TO 800
 630  *CONTINUE
C      HORIZONTAL MOTION PROCESSES HERE. NOT IMPLEMENTED YET.
A      DISPLH "ARG630",2;
      *GO TO 1100
 650  *CONTINUE
      *IF(ARRAY.NE.3)GO TO 700
C      NORSAR DATA. IT IS ARRAHGED VNEVNEVNE... WE DONT HAVE TO DO
C      ANY COMPUTING, JUST REARRANGING.
      *IF(.ANY.((COMP(*).NE.0)))GO TO 690
C       0 MEANS VERTICAL.
C       INDEX1=1,2,3...NSITE (INC 1)
C       INDEX2=1, (INC TWSZR)
C       INDEX3=1, (INC TWSZR*3)
C       INDEX4=1, (INC 3)
        INDEX2=1
        INDEX3=1
        INDEX4=1
       T1=TWSZR+TWSZR+TWSZR
      *DO 680 INDEX1=1,NSITE
       T2=0
       T7=CHGOOD(INDEX4)
      *IF(T7.EQ.0)GO TO 670
       T2=1
      *DO 660 T3=0,TWSZR-1
       T4=INDEX2+T3
       T5=INDEX3+T3
       BUFF2(*,T4,OLD)=BUFF2(*,T5,NEW)
 660  *CONTINUE
       INDEX2=INDEX2+TWSZR
 670  CHGOOD(INDEX1)=T2
      INDEX3=INDEX3+1
      INDEX4=INDEX4+3
 680 *CONTINUE
     *GO TO 800
 690 *CONTINUE
C     HORIZONTAL MOTION. NOT IMPLEMENTED YET.
A     DISPLH "ARG690",2;
     *GO TO 1100
 700 *CONTINUE
C     ALL KNOWN ARRAYS HAVE BEEN CHECKED FOR.
A     DISPLH "ARG700",2;
     *GO TO 1100
 800 *CONTINUE
C     WE NOW HAVE ALL THE DATA CONVERTED TO 64 BIT FLOATING POINT,
C     DEGLITCHED, BIAS REMOVED, VARIANCE CHECKED, MOTION COMPONENTS
C     RESOLVED AND BAD CHANNELS MARKED.CHGOOD(-) IS NOW REALLY
C     SITEGD(-). THE FIRST NSITE ENTRIES ARE THE ONLY ONES WE ARE
C     STILL INTERESTED IN.THEY INDICATE WHICH SITES ARE GOOD. WE NOW
C     SET NGDST TO THE NUMBER OF GOOD SITES, NGDCH TO THE NUMBER
C     OF GOOD CHANNELS (EQUAL TO NGDST FOR VERTICAL MOTION) AND THE
C     VECTOR SITES(-) WILL BE SET SO THAT SITES(I) INDICATES WHICH
C     PHYSICAL SITE LOGICAL SITE I REALLY IS.
      NGDST=0
     *DO 820 INDEX1=1,NSITE
      T7=SITEGD(INDEX1)
     *IF(T7.EQ.0)GO TO 810
      NGDST=NGDST+1
      SITES(NGDST)=INDEX1
 810 *CONTINUE
 820 *CONTINUE
      NGDCH=NGDST
     *IF(.ANY.((COMP(*).EQ.1)))LNGDCH=LNGDCH.SHL.1
     *IF(DEBUG.LT.1)GO TO 830
A     DISPLH "SITEGD",18,SITEGD,SITEGD+70;
A     DISPLH "SITES",16,SITES,SITES+30;
 330 *CONTINUE
C     NOW ITS TIME FOR FFT. FIRST WE HAVE TOCONVERT TO 32-BIT
C     FLOATING POINT, SINCE THATS HOW FFT EXPECTS THE INPUT.
      PINT1(*)=TWSZR*NGDCH
      NGDR=PINT1(1)
     *DO 850 INDEX1=1,NGDR
     *CALL C64T32(BUFF2(*,INDEX1,OLD))
 850 *CONTINUE
C     NOW FFT. NGDCH GIVES THE NUMBER OF FFT'S TO DO.
C     TWSZ GIVES THE SIZE OF EACH FFT.
C     STARTING ADDRESS IS BUFF2(1,1,OLD).
C     ALL THIS IS PASSED IN COMMON TO RUNFFT.
     *CALL RUNFFT
C     NOW TO CONVERT BACK TO 64 BIT FLOATING POINT.
C    *DO 870 INDEX1=1,NGDR
```

```
C    *CALL C32T64(BUFF2(*,INDEX1,OLD))
C870 *CONTINUE
C     NOW WE GO TO BUFF3.
C     FORMAT OF BUFF3 IS (EACH PE):
C       WORD1:          TWTIME          (1 WORD)
C       WORD2:          NGDST           (1 WORD)
C       WORD 3:         SITES(-)        (25 WORDS)
C       WORD 28:        DATA
C               F(LOFREQ)(CH1...CH(NGDST))
C                       .
C                       .
C                       .
C               F(HIFREQ)(CH1...CH(NGDST)
      BF3PE=BF3PE+1
      IBUFF3(BF3PE,1)=TWTIME(1)
      IBUFF3(BF3PE,2)=NGDST
     *DO 920 INDEX1=1,NGDST
      IBUFF3(BF3PE,INDEX1+2)=SITES(INDEX1)
 920 *CONTINUE
     *IF(TWSZ.EG.64)T1=6
     *IF(TWSZ.EQ.128)T1=7
     *IF(TWSZ.EQ.256)T1=8
     *IF(TWSZ.EQ.512)T1=9
      INDEX1=28
      LNGT=LNGDCH.SHL.T1
      T2=0
     *IF(OLD.EQ.2)T2=35200
      T5=LOFREQ
      T6=HIFREQ
     *DO 960 F=T5,T6
C     WE WOULD LIKE TO DO "DO 950 CH=0,NGT-1,TWSZ" BUT CFD INSISTS THAT
C     THE INCREMENT BE CONSTANT SO WE WILL CONSTRUCT AN EQUIVALENT
C     LOOP.
      CH=0
     *DO 950 T7=1,NGDCH
      INDEX2=CH+F+T2
      BUFF3(BF3PE,INDEX1)=RBUFF2(INDEX2)
      INDEX1=INDEX1+1
     *IF(INDEX1.LE.640)GO TO 930
C     BUFF3 IS OVER FLOWING.
A     DISPLH "ARG930",2;
     *GO TO 1100
 930 *CONTINUE
      CH=CH+TWSZ
 950 *CONTINUE
 960 *CONTINUE
     *IF(BF3PE.LT.64)GO TO 1000
C     HAVE TO WRITE OUT BUFF3.
      BF3PE=0
     *WRITE(64,BUFF3(1,1),OUTDM2(OPAGE),40)
     *WAIT 64
      OPAGE=OPAGE+40
     *IF(DEBUG.LT.1)GO TO 965
```

```
A      DISPLH"BUFF3",2;
      *IF(DEBUG.LT.2)GO TO 965
A      DISPLH ,16,BUFF3,BUFF3+1023;
 965 *CONTINUE
C      ZERO OU BUFF3. NOT REALLY REQUIRED, BUT USEFUL ANYWAY.
      *DO 970 INDEX1=1,640
       BUFF3(*,INDEX1)=0.0

 970 *CONTINUE
 1000*CONTINUE
      *IF(.ANY.((TOTSCN(*).NE.FINSCN(*))))GO TO 100
 1100*CONTINUE
C      GOING TO END OF JOB.FIRST WRITE OUT REMNANTS OF BUFF3.
      *WRITE(64,BUFF3(1,1),OUTDM2(OPAGE),40)
      *WAIT 64
      *IF(DEBUG.LT.2)GO TO 1105
A      DISPLH "EOJ",18,BUFF3,BUFF3+1023;
 1105*CONTINUE
A      JUMP EFINALPRINT;
A      BLK;
A      BFINALPRINT::;
A      DATA (("*********")8,ODOA:16)10,"DEM2 GOING TO END OF JOB",
A           (("*********")8,ODOA:16)10;
A      EFINALPRINT::;
A      DISPLS ,16,BFINALPRINT,EFINALPRINT-1;
A      CLSDISP DISP2;
      *CONTINUE
      *STOP
      *END
```

FKCOMB

```
I  <KERR>CFD.FKCOMB;53   THU 22-AUG-74 8:24AM                    PAGE 1

C      FKCOMB
C      WRITTEN FOR ILLIAC BY ANN KERR MAY 1974.
C      PROGRAM READS IN DATA THAT HAS BEEN FFT'D
C      AND ARRANGED WITH ONE TIME WINDOW PER PE AND
C      DETECTS SEISMICEVENTS BY SEARCHING A THREE DIMENSIONAL
C      SPACE,ONE DIMENSION OF FREQUENCY AND TWO DIMENSIONS
C      OF WAVE NUMBER.
C      DECLARATIONS:
     *PE INTEGER INBUF(*,640),CNTRL(*,6),NCHAN(*),PINT1(*),OFFSET(*),    -
     1           LOCATE(*),NPTS(*),COUNT2(*),COUNT3(*),LOC2D(*,25),      -
     2           LOC3D(*,25),TWTIME(*),ADJF(*)
     *PE REAL POWER(*,25),FMAX(*,25),FKX(*,25),FKY(*,25),RINBUF(*,640), -
     1        X(*,25),Y(*,25),FFT(*,612),KERNEL(*,25),                   -
     2        XCOORD(*),YCOORD(*),PREAL1(*),      COSK(*),SINK(*),       -
     4        COSDK(*),SINDK(*),BEAMER(*),FPMAX(*),KXMAX(*),             -
     5        KYMAX(*),DELX(*),DELY(*),KXSEP(*),KYSEP(*),KSEP(*),        -
     6        VEL(*),AZ(*),SIGNAL(*),FSTAT(*),SUMSQ(*)    ,              -
     7        TEST(*),K(*),CHANAV(*),TPOWER(*),FREQ(*)
     *PE REAL ADKX(4),ADKY(4),YPOINT(50),YMAX(50),DX(500),DY(500)
     *PE REAL BEAM(*),TPOW(*),DELTAK(*),P
     *PE REAL PREAL2(*),RPOWER(*,25),IPOWER(*,25),RTPOW(*),ITPOW(*)
     *PE INTEGER MAX
     *CU INTEGER LOFREQ,HIFREQ,DEBUG,SM,T1,T2,ARRAY,PAGE,I,N,MNCHAN,     -
     1           MNPTS,NPOINT,SWITCH,NFREQ,IGO,LINE,LINES,INDEX,IP,      -
     2           TWIN,SAM,IFREQ,J,NFREQ1,REFINE,IND,YTOP        ,        -
     3            YPMI,SIGN,NTIMES,LINEP1
     *PE REAL DELTX(3000),DELTY(3000),DIST
     *PE REAL DELTAX,DELTAY,KX,KY
     *CU REAL DKX,LOWER,UPPER,LINEP,HDKX,BORDER,TWOH
     *CU REAL DELTAF,RADIUS,ANGLE
     *CU LOGICAL MODE3,NMODE
     *EXTERNAL MAX,FNGRID,REALE,IMG,GRID,CHECKR,OUTPUT
     *COMMON/MAINFK/INBUF,CNTRL,NCHAN,PINT1,OFFSET,LOCATE,NPTS,COUNT2,   -
     1              COUNT3,LOC2D,LOC3D,POWER,FMAX,FKX,FKY,X,Y,KERNEL,    -
     2                   XCOORD,YCOORD,PREAL1      ,COSK,SINK,BEAM,      -
     3              TPOW,DELTAK,RPOWER,IPOWER,            COSDK,SINDK,   -
     4              BEAMER,FPMAX,KXMAX,KYMAX,DELX,DELY,KXSEP,KYSEP,KSEP,-
     5              TWTIME,TPOWER,VEL,AZ,SIGNAL,FSTAT,SUMSQ,TEST,K,      -
     6              CHANAV,FREQ,ADJF,DX,DY,P,YPOINT,YMAX,ADKX,ADKY,      -
     7               KX,KY,DELTX,DELTY
     *EQUIVALENCE(INBUF(1,1),RINBUF(1,1)),(INBUF(1,28),FFT(1,1))
     *EQUIVALENCE (1,LOFREQ),(2,HIFREQ),(3,DEBUG),(4,SM),(5,T1),(6,T2), -
     1           (7,ARRAY),(8,PAGE),(9,I),(10,N),        (12,MNCHAN),   -
     2           (13,MNPTS),(14,NPOINT),(15,SWITCH),(16,IGO),            -
     3                                (17,INDEX),(18,IP),(19,DKX),       -
     4           (20,LOWER),(21,UPPER),(22,LINE),(23,LINES),             -
     5                     (24,HDKX),(25,BORDER),(26,TWOH),              -
     6           (27,DELTAF),(28,RADIUS),          (29,SIGN),            -
     7           (30,MODE3),(31,NMODE),(32,TWIN),(33,ANGLE),(34,SAM),    -
                                        **
     8           (35,NFREQ),(36,IFREQ),(37,J),(38,NFREQ1),(39,REFINE)    -
     9           ,(40,NTIMES),(41,IND),(42,YTOP),(43,YPMI),(44,LINEP),   -
     0            (45,LINEP1)
     *DISK AREA CONPRM(1),STCORD(1),FKIN(81)
      MODE=ON
```

```
A      OPNDISP FKDISP;
A      JUMP EHEAD;
A      FKDISP:AREA "FKDISP";
A      BHEAD:BLK;
A      DATA (("********")8,ODOA:16)2,
A      "START EXECUTION FKCOMB",(("********")8,ODOA:16)2;
A      EHEAD:DISPLS,16,BHEAD,EHEAD-1;
      *READ(64,INBUF(1,1),CONPRM(1),1)
      *WAIT 64
       TWIN=INBUF(2,1)
       DKX=RINBUF(12,1)
       LOFREQ=INBUF(6,1)
       HIFREQ=INBUF(7,1)
       LOWER=RINBUF(13,1)
       UPPER=RINBUF(14,1)
       PREAL1(*)=RINBUF(15,1)*0.0174533
       ANGLE=PREAL1(1)
       REFINE=INBUF(16,1)
       SAM=INBUF(17,1)
       NFREQ=HIFREQ-LOFREQ+1
       NFREQ1=NFREQ-1
       DEBUG=INBUF(1,1)
A      SKIP ,ET;
A      T:WDS 1;
A      ET:SLIT(0) T;
A      STORE(0) TWIN;
A      DISPLH "TWIN",16,T,T;
A      STORE(0) DKX;
A      DISPLF "DKX",16,T,T;
A      STORE(0) LOFREQ;
A      DISPLH "LOFREQ",16,T,T;
A      STORE(0) HIFREQ;
A      DISPLH "HIFREQ",16,T,T;
A      STORE(0) LOWER;
A      DISPLF "LOWER",16,T,T;
A      STORE(0) UPPER;
A      DISPLF "UPPER",16,T,T;
A      STORE(0) ANGLE;
A      DISPLH "ANGLE",16,T,T;
A      STORE(0) REFINE;
A      DISPLH "REFINE",16,T,T;
A      STORE(0) SAM;
A      DISPLH "SAM",16,T,T;
A      STORE(0) NFREQ;
A      DISPLH "NFREQ",16,T,T;
A      STORE(0) DEBUG;
A      DISPLH "DEBUG",16,T,T;
      *READ(64,INBUF(1,1),STCORD(1),1)
      *WAIT 64
       XCOORD(*)=RINBUF(*,1)
       YCOORD(*)=RINBUF(*,2)
```

```
A      DISPLF "XCOORD",16,XCOORD,XCOORD+16;
A      DISPLF "YCOORD",16,YCOORD,YCOORD+16;
      *READ(64,INBUF(1,1),FKIN(1),1)
      *WAIT 64
       T1=INBUF(1,1)
      *DO 10 ARRAY=1,7
      *IF(ARRAY.EQ.7)GO TO 9000
C      THAT MEANS UNKNOWN HEADER
       T2=CNTRL(1,ARRAY)
      *IF(T1.EQ.T2)GO TO 15
C      THAT MEANT WE FOUND IT
    10*CONTINUE
    15*CONTINUE
A      SLIT(0) T;
A      STORE(0) ARRAY;
A      DISPLF "ARRAY NO",16,T,T;
       PAGE=2
C A BUNCH OF DEBUG PRINT OUT O F INITIAL VALUES
    20*CONTINUE
    50*CONTINUE
       MODE=ON
      *READ(64,INBUF(1,1),FKIN(PAGE),40)
      *WAIT 64
       PAGE=PAGE+40
       TWTIME(*)=INBUF(*,1)
       NCHAN(*) =INBUF(*,2)
       PINT1(*)=MAX(NCHAN(*))
       MNCHAN=PINT1(1)
      *IF(.ALL.((NCHAN(*).EQ.0))) GO TO 9100
       PINT1(*)=1
      *DO 60 T1=1,25
       MODE=(INBUF(*,PINT1(*)+2).EQ.T1)
       X(*,PINT1(*))=XCOORD(T1)
       Y(*,PINT1(*))=YCOORD(T1)
       PINT1(*)=PINT1(*)+1
       MODE=ON
    60*CONTINUE
      *IF(DEBUG.LT.1)GO TO 70
A      DISPLH "COORDIN",0;
      *IF(DEBUG.LT.2)GO TO 70
A      DISPLF "X-0",16,X,X+3;
A      DISPLF "X-1",16,X+64,X+64+3;
A      DISPLF "X-2",16,X+2*64,X+2*64+3;
A      DISPLF "X-3",16,X+3*64,X+3*64+3;
A      DISPLF "Y-0",16,X,X+3;
A      DISPLF "Y-1",16,Y+64,Y+64+3;
A      DISPLF "Y-2",16,Y+2*64,Y+2*64+3;
A      DISPLF "Y-3",16,Y+3*64,Y+3*64+3;
```

```
A       DISPLH "NCHAN",16,NCHAN,NCHAN+3;
   70*CONTINUE
        OFFSET(*)=0
       *DO 1000 IFREQ=LOFREQ,HIFREQ
        SWITCH =1
       *CALL GRID
        TPOWER(*)=0.0
        RTPOW(*)=0.0
        ITPOW(*)=0.0
        LOCATE(*)=1
       *DO 100 N=1,MNCHAN
        MODE=(N.LE.NCHAN(*))
       *CALL REALE(FFT(*,OFFSET(*)+N),PREAL1(*))
       *CALL IMG  (FFT(*,OFFSET(*)+N),PREAL2(*))
        KERNEL(*,N)=          +(6.28318530*(DELTX(1)*X(*,N)+DELTY(1)*Y(*,N)-
       1))
        COSK(*)=COS(KERNEL(*,N))
        SINK(*)=SIN(KERNEL(*,N))
        RPOWER(*,N)=PREAL1(*)*COSK(*)-PREAL2(*)*SINK(*)
        IPOWER(*,N)=PREAL1(*)*SINK(*)+PREAL2(*)*COSK(*)
        RTPOW(*)=RTPOW(*)+RPOWER(*,N)
        ITPOW(*)=ITPOW(*)+IPOWER(*,N)
  100*CONTINUE
        MODE=ON
        TPOWER(*)=ITPOW(*)**2+RTPOW(*)**2
       *IF(DEBUG.LT.2)GO TO 110
A       DISPLF "FIRST",0;
A       DISPLF "RTPOW",16,RTPOW,RTPOW+3;
A       DISPLF "ITPOW",16,ITPOW,ITPOW+3;
A       DISPLF "TPOWER",16,TPOWER,TPOWER+3;
A       DISPLF "DELTX(1)",16,DELTX,DELTX;
A       DISPLF "DELTY(1)",16,DELTY,DELTY;
  110 *CONTINUE
C       RESTORE MODE
        T1=NPTS(1)
       *DO 300 NPOINT=2,T1
        RTPOW(*)=0.0
        ITPOW(*)=0.0
        MBIT1=MODE
  150 *CONTINUE
       *DO 200 N=1,MNCHAN
        MODE=MODE.AND.(N.LE.NCHAN(*))
        DELTAK(*)=+6.28318530*(DELTX(NPOINT)*X(*,N)+
       1    DELTY(NPOINT)*Y(*,N))                                              -
        COSDK(*)=COS(DELTAK(*))
        SINDK(*)=SIN(DELTAK(*))
        PREAL1(*)=RPOWER(*,N)*COSDK(*)-IPOWER(*,N)*SINDK(*)
        IPOWER(*,N)=RPOWER(*,N)*SINDK(*)+IPOWER(*,N)*COSDK(*)
        RPOWER(*,N)=PREAL1(*)
        RTPOW(*)=RTPOW(*)+RPOWER(*,N)
        ITPOW(*)=ITPOW(*)+IPOWER(*,N)
  200*CONTINUE
      MODE=MBIT1
      TPOW(*)=RTPOW(*)**2+ITPOW(*)**2
C     RESTORE MODE
      MODE=(TPOW(*).GT.TPOWER(*))
      TPOWER(*)=TPOW(*)
      LOCATE(*)=NPOINT
      MODE=MBIT1
  300*CONTINUE
      FPMAX(*)=TPOWER(*)
      MODE=ON
      SWITCH=2
     *CALL GRID
     *DO 350 N=1,MNCHAN
      MODE=(N.LE.NCHAN(*))
     *CALL REALE(FFT(*,OFFSET(*)+N),PREAL1(*))
     *CALL IMG  (FFT(*,OFFSET(*)+N),PREAL2(*))
      KERNEL(*,N)=          +(6.28318530*(KXMAX(*)*X(*,N)+KYMAX(*)*      -
     1              Y(*,N)))
      COSK(*)=COS(KERNEL(*,N))
      SINK(*)=SIN(KERNEL(*,N))
      RPOWER(*,N)=PREAL1(*)*COSK(*)-PREAL2(*)*SINK(*)
      IPOWER(*,N)=PREAL1(*)*SINK(*)+PREAL2(*)*COSK(*)
  350*CONTINUE
      MODE=ON
     *IF(DEBUG.LT.2)GO TO 360
A     DISPLF "COARSE",0;
A     DISPLH "NPTS",16,NPTS,NPTS;
A     DISPLH "LOCATE",16,LOCATE,LOCATE+3;
A     DISPLF "KXMAX",16,KXMAX,KXMAX+3;
A     DISPLF "KYMAX",16,KYMAX,KYMAX+3;
      PREAL1(*)=RPOWER(*,N)
      PREAL2(*)=IPOWER(*,N)
A     DISPLF "RPOWER",16,PREAL1,PREAL1+3;
A     DISPLF "IPOWER",16,PREAL2,PREAL2+3;
A     DISPLF "FPMAX",16,FPMAX,FPMAX+3;
 360 *CONTINUE
     *CALL FNGRID
      FMAX(*,IFREQ)=FPMAX(*)
      FKX(*,IFREQ)=KXMAX(*)
      FKY(*,IFREQ)=KYMAX(*)
      OFFSET(*)=OFFSET(*)+NCHAN(*)
 1000*CONTINUE
     *CALL CHECKR
C DISPLAY MAXIMUM AND ASSORTED PARAMETERS
     *CALL OUTPUT
     *GO TO 50
 9000*CONTINUE
A     DISPLH "BAD HEAD",0;
 9100*CONTINUE
A     JUMP PBANNER;
```

```
A      BANNER:BLK;
A      DATA (("********")8,ODOA:16)2,"FKCOMB EOJ",
A      (("********")8,ODOA:16)2;
A      PBANNER:DISPLS ,16,BANNER,PBANNER-1;
A      CLSDISP FKDISP;
      *STOP

      *END
```

SUBROUTINES

```
*BLOCK DATA
*PE INTEGER INBUF(*,640),CNTRL(*,6),NCHAN(*),PINT1(*),OFFSET(*),       -
1             LOCATE(*),NPTS(*),COUNT2(*),COUNT3(*),LOC2D(*,25),       -
2             LOC3D(*,25),TWTIME(*),ADJF(*)
*PE REAL POWER(*,25),FMAX(*,25),FKX(*,25),FKY(*,25),RINBUF(*,640),     -
1        X(*,25),Y(*,25),FFT(*,612),KERNEL(*,25),                      -
2        XCOORD(*),YCOORD(*),PREAL1(*),      COSK(*),SINK(*),          -
4        COSDK(*),SINDK(*),BEAMER(*),FPMAX(*),KXMAX(*),                -
5        KYMAX(*),DELX(*),DELY(*),KXSEP(*),KYSEP(*),KSEP(*),           -
6        VEL(*),AZ(*),SIGNAL(*),FSTAT(*),SUMSQ(*)     ,                -
7        TEST(*),K(*),CHANAV(*),TPOWER(*),FREQ(*)
*PE REAL ADKX(4),ADKY(4),YPOINT(50),YMAX(50),DX(500),DY(500)
*PE REAL BEAM(*),TPOW(*),DELTAK(*),P
*PE REAL PREAL2(*),RPOWER(*,25),IPOWER(*,25),RTPOW(*),ITPOW(*)
*PE INTEGER MAX
*PE REAL DELTX(3000),DELTY(3000),DIST
*PE REAL DELTAX,DELTAY,KX,KY
*COMMON/MAINFK/INBUF,CNTRL,NCHAN,PINT1,OFFSET,LOCATE,NPTS,COUNT2,      -
1               COUNT3,LOC2D,LOC3D,POWER,FMAX,FKX,FKY,X,Y,KERNEL,      -
2                   XCOORD,YCOORD,PREAL1       ,COSK,SINK,BEAM,        -
3               TPOW,DELTAK,RPOWER,IPOWER,             COSDK,SINDK,    -
4               BEAMER,FPMAX,KXMAX,KYMAX,DELX,DELY,KXSEP,KYSEP,KSEP,-
5               TWTIME,TPOWER,VEL,AZ,SIGNAL,FSTAT,SUMSQ,TEST,K,        -
6               CHANAV,FREQ,ADJF,DX,DY,P,YPOINT,YMAX,ADKX,ADKY,        -
7                KX,KY,DELTX,DELTY
*EQUIVALENCE(INBUF(1,1),RINBUF(1,1)),(INBUF(1,28),FFT(1,1))
*DATA CNTRL/54227,63*0,49619,63*0,60366,63*0,192*0/
*END
```

```
*BLOCK DATA
*PE INTEGER CNTRL(*,6),OUTBUF(*,64,6),PINTI(*),INBUF(*,128),             -
1          TIME(*),OLDTIM(*),                                             -
1                     SAVBCT,SAVPTW,OUPAGE(6),TSTEPS(6),SCANS,            -
2           OUPTWA(6),      OTIMEA(6),ORGADB,INBUF1(8192)
*COMMON/MAIN/CNTRL,OUTBUF,INBUF,PINTI,TIME,OLDTIM,SAVBCT,SAVPTW,          -
1      TSTEPS,SCANS,OUPTWA,OUPAGE,OTIMEA,ORGADB
*EQUIVALENCE (INBUF(1,1),INBUF1(1))
*DATA OUPAGE/6*2/
*DATA CNTRL/1,1,51,16,1,0,103,446,10,1,54227,53*0,                        -
1           0,2,57,0,0,6,29,95,15,2,49619,53*0,                           -
2           0,1,66,0,1,72,12,0,10,1,59366,53*0,                           -
3          10*0,65536,53*0,10*0,65536,53*0,10*0,65536,53*0/
*END
```

```
*BLOCK DATA
*PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),     -
1           PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
*PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
1        PREAL2(*),ALLMSQ(*),TVARFT(*)
*PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
1           CHGOOD(80),SITEGD(80),SITES(80)
*PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
*COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,   -
1       TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,    -
2       ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
*EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),      -
1             ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),          -
2           (BUFF3(1,1),IBUFF3(1,1))
*DATA PEN/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,      -
1         21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,   -
2         39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,   -
3         57,58,59,60,61,62,63,64/
*DATA CNTRL/54227,0,51,17,60*0,                                    -
1           49619,0,57,19,60*0,                                    -
2           60366,0,66,22,60*0,                                    -
3           65536,63*0,65536,63*0,65536,63*0/
*END
```

Best Available Copy

for page 29

```
      *SUBROUTINE CHECKR
C      PURPOSE:VERIFY TWO AND THREE DIMENSIONAL MAXIMUM.TEST EACH
C      POWER MAXIMUM FOR EACH FREQUENCY.
C      DECLARATIONS:
      *PE INTEGER INBUF(*,640),CNTRL(*,6),NCHAN(*),PINT1(*),OFFSET(*),     -
      1           LOCATE(*),NPTS(*),COUNT2(*),COUNT3(*),LOC2D(*,25),        -
      2           LOC3D(*,25),TWTIME(*),ADJF(*),INDEX(*)
      *PE REAL POWER(*,25),FMAX(*,25),FKX(*,25),FKY(*,25),RINBUF(*,640),   -
      1        X(*,25),Y(*,25),FFT(*,612),KERNEL(*,25),                    -
      2        XCOORD(*),YCOORD(*),PREAL1(*),       COSK(*),SINK(*),       -
      4        COSDK(*),SINDK(*),BEAMER(*),FPMAX(*),KXMAX(*),              -
      5        KYMAX(*),DELX(*),DELY(*),KXSEP(*),KYSEP(*),KSEP(*),         -
      6        VEL(*),AZ(*), IGNAL(*),FSTAT(*),SUMSQ(*)     ,              -
      7        TEST(*),K(*),CHANAV(*),TPOWER(*),FREQ(*)
      *PE REAL ADKX(4),ADKY(4),YPOINT(50),YMAX(50),DX(500),DY(500)
      *PE REAL BEAM(*),TPOW(*),DELTAK(*),CHECK(*,3),P
      *PE REAL PREAL2(*),RPOWER(*,25),IPOWER(*,25),RTPOW(*),ITPOW(*)
      *PE INTEGER MAX
      *CU INTEGER LOFREQ,HIFREQ,DEBUG,SM,T1,T2,ARRAY,PAGE,I,N,MNCHAN,       -
      1           MNPTS,NPOINT,SWITCH,NFREQ,IGO,LINE,LINES       ,IP,       -
      2           TWIN,SAM,IFREQ,J,NFREQ1,REFINE,IND,YTOP        ,          -
      3            YPMI,SIGN,NTIMES,LINEP1
      *PE REAL DELTX(3000),DELTY(3000),DIST
      *PE REAL DELTAX,DELTAY,KX,KY
      *CU REAL DKX,LOWER,UPPER,LINEP,HDKX,BORDER,TWOH
      *CU REAL DELTAF,RADIUS,ANGLE
      *CU LOGICAL MODE3,NMODE
      *EXTERNAL MAX,FNGRID,REALE,IMG,GRID
      *COMMON/MAINFK/INBUF,CNTRL,NCHAN,PINT1,OFFSET,LOCATE,NPTS,COUNT2,   -
      1              COUNT3,LOC2D,LOC3D,POWER,FMAX,FKX,FKY,X,Y,KERNEL,     -
      2                     XCOORD,YCOORD,PREAL1       ,COSK,SINK,BEAM,    -
      3              TPOW,DELTAK,RPOWER,IPOWER,               COSDK,SINDK, -
      4              BEAMER,FPMAX,KXMAX,KYMAX,DELX,DELY,KXSEP,KYSEP,KSEP,-
      5              TWTIME,TPOWER,VEL,AZ,SIGNAL,FSTAT,SUMSQ,TEST,K,       -
      6              CHANAV,FREQ,ADJF,DX,DY,P,YPOINT,YMAX,ADKX,ADKY,       -
      7               KX,KY
      *EQUIVALENCE(INBUF(1,1),RINBUF(1,1)),(INBUF(1,28),FFT(1,1))
      *EQUIVALENCE (1,LOFREQ),(2,HIFREQ),(3,DEBUG),(4,SM),(5,T1),(6,T2), -
      1            (7,ARRAY),(8,PAGE),(9,I),(10,N),          (12,MNCHAN), -
      2            (13,MNPTS),(14,NPOINT),(15,SWITCH),(16,IGO),           -
      3                                          (18,IP),(19,DKX),        -
      4            (20,LOWER),(21,UPPER),(22,LINE),(23,LINES),            -
      5                        (24,HDKX),(25,BORDER),(26,TWOH),           -
      6            (27,DELTAF),(28,RADIUS),             (29,SIGN),        -
      7            (30,MODE3),(31,NMODE),(32,TWIN),(33,ANGLE),(34,SAM),   -
                                         **
      8            (35,NFREQ),(36,IFREQ),(37,J),(38,NFREQ1),(39,REFINE)   -
      9            ,(40,NTIMES),(41,IND),(42,YTOP),(43,YPMI),(44,LINEP),  -
      0             (45,LINEP1)
      *DISK AREA CONPRM(1),STCORD(1),FKIN(81)
      *IF(DEBUG.LT.1)GO TO 10
```

```
A       DISPLH "CHECKR",0;
  10 *CONTINUE
        COUNT2(*)=0
        COUNT3(*)=0
        PREAL1(*)=0.5*DKX
        HDKX=PREAL1(1)
       *DO 100 I=LOFREQ+1,HIFREQ-1
        MODE3=(FMAX(*,I).GT.FMAX(*,I-1))
        MBIT1=(FMAX(*,I).GT.FMAX(*,I+1))
        MODE3=MODE3.AND.MBIT1
        MODE=MODE.AND..NOT.MODE3
       *IF((FMAX(*,I-1).GT.FMAX(*,I))) INDEX(*)=I-1
       *IF(( FMAX(*,I+1).GT.FMAX(*,I))) INDEX(*)=I+1
        KXSEP(*)=FKX(*,I)-FKX(*,INDEX(*))
        KYSEP(*)=FKY(*,I)-FKY(*,INDEX(*))
        KSEP(*)=SQRT(KXSEP(*)**2+KYSEP(*)**2)
        NMODE=(KSEP(*).LT.HDKX)
        MODE=MODE.AND..NOT.NMODE
        ADJF(*)=I-1
       *DO 560 J=1,2
        DELY(*)=FKY(*,ADJF(*))-FKY(*,I)
        DELX(*)=FKX(*,ADJF(*))-FKX(*,I)
        MBIT2=MODE
        OFFSET(*)=(ADJF(*)-LOFREQ)*NCHAN(*)
        RTPOW(*)=0.0
        ITPOW(*)=0.0
       *DO 550 N=1,MNCHAN
        MODE=MODE.AND.(N.LE.NCHAN(*))
       *CALL REALE(FFT(*,OFFSET(*)+N),PREAL1(*))
       *CALL IMG  (FFT(*,OFFSET(*)+N),PREAL2(*))
        DELTAK(*)=          +(6.28318530*(FKX(*,ADJF(*))*X(*,N)+
       1      FKY(*,ADJF(*))*Y(*,N)))
        COSK(*)=COS(DELTAK(*))
        SINK(*)=SIN(DELTAK(*))
        RTPOW(*)=PREAL1(*)*COSK(*)-PREAL2(*)*SINK(*)+RTPOW(*)
        ITPOW(*)=PREAL1(*)*SINK(*)+PREAL2(*)*COSK(*)+ITPOW(*)
  550*CONTINUE
        MODE=MBIT2
        CHECK(*,J)=RTPOW(*)**2+ITPOW(*)**2
        ADJF(*)=I+1
  560*CONTINUE
        MODE=ON
        MBIT2=(FMAX(*,I).GT.CHECK(*,1))
        MBIT1=(FMAX(*,I).GT.CHECK(*,2))
        MBIT1=MBIT1.AND.MBIT2
        MBIT1=MBIT1.AND..NOT.NMODE
        MODE3=MODE3.OR.MBIT1
        NMODE=.NOT.MODE3
        MODE=NMODE
        COUNT2(*)=C UNT2(*)+1
        LOC2D(*,COUNT2(*))=I
        MODE=MODE3
        COUNT3(*)=COUNT3(*)+1
        LOC3D(*,COUNT3(*))=I
        MODE=ON
```

```
A      SETC(0) G;
A      SETC(1) H;
A      DISPLH "MODES",1;
A      DISPLF "CHECK1",16,CHECK,CHECK+9;
A      DISPLF "CHECK2",16,CHECK+64,CHECK+73;
  100* CONTINUE
     *RETURN
     *END
     *SUBROUTINE C16T64(IN,OUT)
     *PE REAL IN(*),OUT(*)
     *PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),    -
     1           PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
     *PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
     1       PREAL2(*),ALLMSQ(*),TVARFT(*)
     *PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
     1           CHGOOD(80),SITEGD(80),SITES(80)
     *PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
     *CU INTEGER ADBBUF(8),COREPT,      BYTE,ADBWRD,ARRAY,DEBUG,TWSZ,   -
     1           OVLAP,NCHAN,NSITE,NROWS,DIFFR,DIFFW,NEW,OLD,GAP,TSCANS,-
     2          INDEX1,INDEX2,INDEX3,INDEX4,T1,T2,T3,T4,T5,T6,CH,       -
     3          OFFSET,INBYT,NGDCH,TWSZR,NGDST,NGDR,F,BF3PE,NGT,PAGE,T7
     *CU LOGICAL LADBBU(8),LCOREP,LAST16,LBYTE,LADBWR,LARRAY,LDEBUG,    -
     1           LTWSZ,LOVLAP,LNCHAN,LNSITE,LNROWS,LDIFFR,LDIFFW,LNEW,  -
     2           LOLD,LGAP,LTSCAN ,LT1,LT2,LT3,LT4,LT5,LT6,LCH,LOFFSE,  -
     3           LINBYT,LF,LNGDCH,LTWSZR,LNGDST,LNGDR,LNGT,LT7
     *EXTERNAL GTDATA,C64T32,ROWSUM,RUNFFT,C32T64
     *COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,   -
     1      TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,     -
     2      ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
     *EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),      -
     1              ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),          -
     2           (BUFF3(1,1),IBUFF3(1,1))
     *EQUIVALENCE (1,ADBBUF(1),LADBBU(1)),(9,COREPT,LCOREP),(10,BYTE,   -
     1           LBYTE),(11,ADBWRD,LADBWR),(12,ARRAY,LARRAY),(13,TWSZ,  -
     2           LTWSZ),(14,OVLAP,LOVLAP),(15,NCHAN,LNCHAN),(16,NSITE,  -
     3           LNSITE),(17,NROWS,LNROWS),(18,DIFFR,LDIFFR),(19,DIFFW, -
     4           LDIFFW),(20,NEW,LNEW),(21,OLD,LOLD),(22,GAP,LGAP),(23, -
     5           TSCANS,LTSCAN),(24,INDEX1),(25,INDEX2),(26,INDEX3),(27,-
     6           INDEX4),(28,T1,LT4),(29,T2,LT2),(30,T3,LT3),(31,T4,    -
     7           LT4),(32,T5,LT5),(33,T6,LT6),(34,CH,LCH),(35,OFFSET,   -
     8           LOFFSE),(36,INBYT,LINBYT),(37,F,LF),(38,NGDCH,LNGDCH), -
     9           (39,TWSZR,LTWSZR),(40,NGDST,LNGDST),(41,NGDR,LNGDR),   -
     0           (42,BF3PE),(43,NGT,LNGT),(44,LAST16),(45,DEBUG,LDEBUG),-
     1           (46,PAGE),(47,T7,LT7)
     *DISK AREA INDM2(20),OUTDM2(40),CONPRM(1)
     *IF(DEBUG.LT.1)GO TO 10
A     DISPLH "C16T64",0;
     *IF(DEBUG.LT.3)GO TO 10
A     LDL(0) $D49;
A     LDA IN(0);
A     DISPLH "IN",32;
  10 *CONTINUE
C     NOW TO CHECK WHICH ARRAY WE HAVE.
     *IF(ARRAY.GT.2)GO TO 100
C     LASA AND ALPA HERE. SIMPLE 14 BIT TWOS COMPLEMENT.
A     LDL(0) $D49;
A     LDA IN(0);
A     SHAR =2;             % RIGHT JUSTIFY IT.
A     AND =3FFF:16;               % GOT RID OF ANY GARBAGE BITS.
```

```
A      LDR $A:                      % SAVE IT IN $R.
A      % NOW TO TAKE CARE OF THE SIGN.
A      LIT(O) =0C00E000000000000:16:
A                                   % WITH A 1 IN THE HIGH ORDER BIT OT THE
A                                   % MANTISSA THATS -(2**13) WHICH IS THE
A                                   % VALUE OF THE SIGN BIT.
A      SHAR =13:                    % ISOLATE THE SIGN BIT.
A      SHAL =47:                    %PUT IT IN THE HIGH ORDER BIT OF THE
A                                   % MANTISSA.
A      LEX $CO:
A      NORM:                        % NOW ITS EITHER -(2**13) OR 0.0  .
A      SAN:
A      LDS $A:                      % SAVE SIGN IN $S.
A      LDA $R:                      % GET THE ORIGINAL.
A      SHAL=35:                     % SIGN BIT WENT IN EXPONENT SOON TO BE
A                                   % OVERWRITTEN.
A      LIT(O) =0400D000000000000:16:
A      LEX $CO:
A      NORM:                        % ABSOLUTE VALUE IS IN FLOATING FORMAT.
A      ADRN $S:                     % ADD IN THE SIGN.
A      STA OUT:                     % GOT IT.
      *GO TO 500
 100 *IF(ARRAY.NE.3)GO TO 200
C      NORSAR DATA. 4 BITS OF GAIN CODE AND 12-BIT TWOS COMPLEMENT
C      MANTISSA.
A      LDL(O) $D49:
A      LDA IN(O):
A      LDS $A:                      % SAVE IT IN $S.
A                                   % FIRST LETS DO THE SIGN.
A      AND=0800:16:
A      SHAL=36:                     % PUT THE SIGN BIT IN THE HIGH ORDER BIT
A                                   % OF THE MANTISSA.
A      LIT(O) =0C00C000000000000:16:
A      LEX $CO:                     % NOW ITS EITHER -(2**11) OR 0.0  .
A      NORM:
A      SAN:
A      LDR $A:                      % SAVE IT IN $R:
A      LDA $S:                      % RESTORE THE ORIGINAL TO DO THE REST
A                                   % OF THE MANTISSA.
A      AND=7FF:16:                  % LOW ORDER 11 BITS.
A       SHAL =37:                   % LEFT JUSTIFY THEM IN THE MANTISSA.
A      LIT(O) =0400B00000000000:16:
A      LEX $CO:
A      NORM:                        % THATS THE MANTISSA WITHOUT THE SIGN.
A      ADRN $R:                     % THATS THE MANTISSA WITH THE SIGN.
A      LDR $A:                      % SAVE IT IN $R.
A      LDA $S:                      % RESTORE THE ORIGINAL TO DO GAIN CODE.
A      SHAR =12:                    % ISOLATE THE 4 PITS OF GAIN CODE.
A      SAN:                         % MAKE IT NEGATIVE.
A      ADM =400C:16:                % WE WANT EXPONENT TO BE THAT OF
A                                   % 2**(10-GAIN). THATS EXPONENT OT 2**10.
```

```
A      SHAL =48;
A      SAB =16;                  % THATS GOT THE GAIN CODE.
A      MLRN $R;                  % MULTIPLY IN THE MANTISSA AND SIGN.
A      LDL(0) $D50;
A      STA OUT(0);
      *GO TO 500
 200  *CONTINUE
A      DISPLH "ARG200",2;
C '    ARRAY WAS OUT OF RANGE.
 500  *CONTINUE
      *IF(DEBUG.LT.1)GO TO 510
A      DISPLH "EC16T64",0;
      *IF(DEBUG.LT.3)GO TO 510
A      LDL(0) $D50;
A      STA OUT(0);
 510  *CONTINUE
      *RETURN
      *END
      *SUBROUTINE C32T64(IN,OUT)
      *PE REAL IN(*),OUT(*)
      *PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),      -
      1           PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
      *PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
      1        PREAL2(*),ALLMSQ(*),TVARFT(*)
      *PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
      1           CHGOOD(80),SITEGD(80),SITES(80)
      *PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
      *CU INTEGER ADBBUF(8),COREPT,      BYTE,ADBWRD,ARRAY,DEBUG,TWSZ,    -
      1           OVLAP,NCHAN,NSITE,NROWS,DIFFR,DIFFW,NEW,OLD,GAP,TSCANS,-
      2          INDEX1,INDEX2,INDEX3,INDEX4,T1,T2,T3,T4,T5,T6,CH,          -
      3          OFFSET,INBYT,NGDCH,TWSZR,NGDST,NGDR,F,BF3PE,NGT,PAGE,T7
      *CU LOGICAL LADBBU(8),LCOREP,LAST16,LBYTE,LADBWR,LARRAY,LDEBUG,     -
      1           LTWSZ,LOVLAP,LNCHAN,LNSITE,LNROWS,LDIFFR,LDIFFW,LNEW,    -
      2           LOLD,LGAP,LTSCAN ,LT1,LT2,LT3,LT4,LT5,LT6,LCH,LOFFSE,    -
      3           LINBYT,LF,LNGDCH,LTWSZR,LNGDST,LNGDR,LNGT,LT7
      *EXTERNAL GTDATA,C16T64,C64T32,ROWSUM,RUNFFT
      *COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,    -
      1       TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,     -
      2       ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
      *EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),       -
      1             ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),           -
      2           (BUFF3(1,1),IBUFF3(1,1))
      *EQUIVALENCE (1,ADBBUF(1),LADBBU(1)),(9,COREPT,LCOREP),(10,BYTE,    -
      1           LBYTE),(11,ADBWRD,LADBWR),(12,ARRAY,LARRAY),(13,TWSZ,    -
      2           LTWSZ),(14,OVLAP,LOVLAP),(15,NCHAN,LNCHAN),(16,NSITE,    -
      3           LNSITE),(17,NROWS,LNROWS),(18,DIFFR,LDIFFR),(19,DIFFW,   -
      4           LDIFFW),(20,NEW,LNEW),(21,OLD,LOLD),(22,GAP,LGAP),(23,   -
      5           TSCANS,LTSCAN),(24,INDEX1),(25,INDEX2),(26,INDEX3),(27,-
      6           INDEX4),(28,T1,LT1),(29,T2,LT2),(30,T3,LT3),(31,T4,      -
      7           LT4),(32,T5,LT5),(33,T6,LT6),(34,CH,LCH),(35,OFFSET,     -
      8           LOFFSE),(36,INBYT,LINBYT),(37,F,LF),(38,NGDCH,LNGDCH),   -
      9           (39,TWSZR,LTWSZR),(40,NGDST,LNGDST),(41,NGDR,LNGDR),     -
      0           (42,BF3PE),(43,NGT,LNGT),(44,LAST16),(45,DEBUG,LDEBUG),-
      1           (46,PAGE),(47,T7,LT7)
      *DISK AREA INDM2(20),OUTDM2(40),CONPRM(1)
      *IF(DEBUG.LT.1)GO TO 10
A      DISPLH "C32T64",0;
      *IF(DEBUG.LT.3)GO TO 10
A      LDL(0) $D49;
A      LDA IN(0);
A      DISPLH "IN",32;
   10 *CONTINUE
A      LDL(0) $D49;
A      LDA IN(0);
A      LDR $A;                    % SAVE IT.
A      RAB =0;                    % ELIMINATE THE SIGN SO WE CAN DO THE
A                                 % EXPONENT.
A      SHAR =56;                  % ISOLATE THE EXPONENT.
A      SBM =40;16;                % SUBTRACT THE 32 BIT OFFSET.
```

```
A      ADM =4000:16;              % ADD THE 64 BIT OFFSET.
A      SHAL =48;                  % PUT IT IN THE 64 BIT EXP. FIELD.
A      LDS $A;                    % SAVE IT.
A      LDA $R;                    % NOW FOR THE SIGN.
A      SHAR =63;
A      SHAL =63;                  % SIGN BIT IS ISOLATED.
A      OR $S;                     % NOW WE HAVE EXPONENT AND SIGN.
A      LDS $A;                    % SAVE IT.
A      LDA $R;                    % NOW FOR THE MANTISSA.
A      SHAL =40;                  % ISOLATE THE MANTISSA.
A      SHAR =16;                  % PUT IT IN 64 BIT EXP FIELD.
A      OR $S;                     % DONE.
A      LDL(O) $D50;
A      STA OUT(O);
      *IF(DEBUG.LT.1)GO TO 110
A      DISPLH "EC32T64",0;
      *IF(DEBUG.LT.3)GO TO 110
A      LDL(O) $D50;
A      LDA OUT(O);
A      DISPLH "OUT",32;
 110  *CONTINUE
      *RETURN
      *END
```

```
     *SUBROUTINE C64T32(IN)
     *PE REAL IN(*)
     *PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),     -
     1            PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
     *PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
     1        PREAL2(*),ALLMSQ(*),TVARFT(*)
     *PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
     1            CHGOOD(80),SITEGD(80),SITES(80)
     *PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
     *CU INTEGER ADBBUF(8),COREPT,       BYTE,ADBWRD,ARRAY,DEBUG,TWSZ,   -
     1            OVLAP,NCHAN,NSITE,NROWS,DIFFR,DIFFW,NEW,OLD,GAP,TSCANS,-
     2           INDEX1,INDEX2,INDEX3,INDEX4,T1,T2,T3,T4,T5,T6,CH,       -
     3           OFFSET,INBYT,NGDCH,TWSZR,NGDST,NGDR,F,BF3PE,NGT,PAGE,T7
     *CU LOGICAL LADBBU(8),LCOREP,LAST16,LBYTE,LADBWR,LARRAY,LDEBUG,     -
     1            LTWSZ,LOVLAP,LNCHAN,LNSITE,LNROWS,LDIFFR,LDIFFW,LNEW,  -
     2            LOLD,LGAP,LTSCAN ,LT1,LT2,LT3,LT4,LT5,LT6,LCH,LOFFSE,  -
     3            LINBYT,LF,LNGDCH,LTWSZR,LNGDST,LNGDR,LNGT,LT7
     *EXTERNAL GTDATA,C16T64,ROWSUM,RUNFFT,C32T64
     *COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,    -
     !        TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,    -
     2        ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
     *EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),      -
     1              ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),          -
     2            (BUFF3(1,1),IBUFF3(1,1))
     *EQUIVALENCE (1,ADBBUF(1),LADBBU(1)),(9,COREPT,LCOREP),(10,BYTE,    -
     1            LBYTE),(11,ADBWRD,LADBWR),(12,ARRAY,LARRAY),(13,TWSZ,  -
     2            LTWSZ),(14,OVLAP,LOVLAP),(15,NCHAN,LNCHAN),(16,NSITE,  -
     3            LNSITE),(17,NROWS,LNROWS),(18,DIFFR,LDIFFR),(19,DIFFW, -
     4            LDIFFW),(20,NEW,LNEW),(21,OLD,LOLD),(22,GAP,LGAP),(23, -
     5            TSCANS,LTSCAN),(24,INDEX1),(25,INDEX2),(26,INDEX3),(27,-
     6            INDEX4),(28,T1,LT1),(29,T2,LT2),(30,T3,LT3),(31,T4,    -
     7            LT4),(32,T5,LT5),(33,T6,LT6),(34,CH,LCH),(35,OFFSET,   -
     8            LOFFSE),(36,INBYT,LINBYT),(37,F,LF),(38,NGDCH,LNGDCH), -
     9            (39,TWSZR,LTWSZR),(40,NGDST,LNGDST),(41,NGDR,LNGDR),   -
     0            (42,BF3PE),(43,NGT,LNGT),(44,LAST16),(45,DEBUG,LDEBUG),-
     1            (46,PAGE),(47,T7,LT7)
     *DISK AREA INDM2(20),OUTDM2(40),CONPRM(1)
     *IF(DEBUG.LT.1)GO TO 10
A     DISPLH "C64T32",0;
     *IF(DEBUG.LT.3)GO TO 10
A     LDL(0) $D49;
A     LDA IN(0);
A     DISPLH "IN",32;
A     DISPLF ,32;
  10 *CONTINUE
A     LDL(0) $D49;
A     LDA IN(0);
A     LDS $A;                      % SAVE IT.
A     RAB =0;                      % GET RID OF THE SIGN BIT FOR NOW.
A     SHAR =48;                    % ISOLATE THE EXPONENT.
A     SBM =4000:16;                % SUBTRACT OUT THE 64-BIT OFFSET.
```

```
A      ADM =40:16;                  % ADD IN THE 32-BIT OFFSET.
A      SHAL =56;                    % PUT IT IN 32-BIT OUTER EXPONENT FIELD.
A      LDR $A;                      % SAVE IT.
A      LDA $S;                      % NOW FOR THE SIGN BIT.
A      SHAR =63;
A      SHAL =63;
A      OR $R;                       % $A HAS EXPONENT AND SIGN.
A      LDR $A;                      % SAVE IT.
A      LDA $S;                      % NOW FOR THE MANTISSA.
A      SHAL =16;
A      SHAR =40;                    %THE 32-BIT MANTISSA FIELD.
A      OR $R;                       % THATS ALL OF IT.
A      LDL(O) $D49;
A      STA IN(O);
      *IF(DEBUG.LT.1)GO TO 110
A      DISPLH "EC64T32",0;
      *IF(DEBUG.LT.3)GO TO 110
A      LDL(O) $D49;
A      LDA IN(O);
A      DISPLH "OUT(IN)",32;
 110  *CONTINUE
      *RETURN
      *END
      *SUBROUTINE CNVTIM
      *PE INTEGER CNTRL(*,6),OUTBUF(*,64,6),PINTI(*),INBUF(*,128),         -
      1          TIME(*),OLDTIM(*),                                         -
      1                       SAVBCT,SAVPTW,OUPAGE(6),TSTEPS(6),SCANS,      -
      2           OUPTWA(6),      OTIMEA(6)
      *CU INTEGER ADBBUF(8),ARRAY,INPTB,INPTW,SAVADB,ADBOUT(6),OUPTW,      -
      1           BYTS,WORDS,T1,T2,T3,T4,T5,T6,       IT,PRTIAL,ADDRS,      -
      2           WORD,      BYTCNT(6),ADBWRD,INBYT,OUBYT,ORGCOR,PAGE,      -
      3           DEBUG,BCT,ADB,ENDADB
      *CU LOGICAL LADBBU(8),LARRAY,LINPTB,LINPTW,LSAVAD,LADBOU(6),LOUPTW,-
      1           LBYTS,LWORDS,LT1,LT2,LT3,LT4,LT5,LT6,LOUBYT,LIT,LPRTIA,-
      2           LADDRS,LWORD,LINBYT,LBYTCN(6),LADBWR,LORGCO,LPAGE         -
      3           ,LDEBUG,LBCT,LADB,LENDAD
      *EXTERNAL RDPRM,GETBYT,PUTBYT
      *COMMON/MAIN/CNTRL,OUTBUF,INBUF,PINTI,TIME,OLDTIM,SAVBCT,SAVPTW,      -
      1     TSTEPS,SCANS,OUPTWA,OUPAGE,       OTIMEA
      *EQUIVALENCE(1,ADBBUF(1),LADBBU(1)),(9,ARRAY,LARRAY),                 -
      1           (10,INPTB,LINPTB),                                        -
      1           (11,INPTW,LINPTW),(12,SAVADB,LSAVAD),                     -
      1           (13,ADBOUT(6),LADBOU(6))                                  -
      2           ,(19,OUPTW ,LOUPTW),(20,BYTS,LBYTS),(21,WORDS,LWORDS),    -
      3           (22,T1,LT1),(23,T2,LT2),(24,T3,LT3),(25,T4,LT4),          -
      4           (26,T5,LT5),(27,T6,LT6),(28,OUBYT,LOUBYT),(29,IT,LIT),    -
      5           (30,PRTIAL,LPRTIA),(31,ADDRS,LADDRS),(32,WORD,LWORD),     -
      6           (33,INBYT,LINBYT),(34,BYTCNT(1),LBYTCN(1)),               -
      6           (40,ADBWRD,LADBWR)                                        -
      7                                                                     -
      8                                           ,(43,ORGCOR,LORGCO),-
      9           (44,PAGE,LPAGE),(45,DEBUG,LDEBUG),(46,BCT,LBCT),          -
                  (47,ADB,LADB),(48,ENDADB,LENDAD)
      *DISK AREA OUPUT1(20),OUPUT2(20),OUPUT3(20),OUPUT4(20),OUPUT5(20), -
      1         OUPUT6(20),INPUT(50)
      *IF(DEBUG.LT.1)GO TO 10
A      DISPLH "CNVTIM:",0:
   10  T6=CNTRL(2,ARRAY)
      *IF(T6.NE.1)GO TO 100
C      CNTRL(2,ARRAY) EQUALS 1 ONLY IF TIME IS ALREADY IN THE FORM OF
C      DECISECONDS FROM THE BEGINNING OF THE YEAR AS READ FROM THE TAPE.
C      NO CONVERSION IS NECCESSARY.
C      WE NEED ONLY TO MOVE IT.
      *CALL GETBYT
A      LDL(0) LINBYT:
A      CSHL(0) 16:
A      LDA $C0:
A      STA TIME:
      *CALL GETBYT
A      LDL(0) LINBYT:
A      LDA $C0:
A      OR TIME:
A      STA TIME:
      *IF(DEBUG.LT.1)GO TO 20
A      DISPLH "FMT#1",0:
      *IF(DEBUG.LT.2)GO TO 20
```

```
A       DISPLH "TIME",2:
  20  *GO TO 500
 100   T6=CNTRL(2,ARRAY)
      *IF(T6.NE.2)GO TO 200
C      TIME IS IN EBCDIC IN THE FORM DDDHHMMSS. GET IT A CHARACTER AT A
C      TIME.
      *CALL GETBYT
       LT6=OFF.TURN ON..LAST.4
       LT1=LINBYT.SHR.12
       LT2=LINBYT.SHR.8
       LT2=LT2.AND.LT6
       LT3=LINBYT.SHR.4
       LT3=LT3.AND.LT6
       LT4=LINBYT.AND.LT6
A      %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       TIME(*)=(T1*100+T2*10+T3)*24+T4*10
      *CALL GETBYT
       LT6=OFF.TURN ON..LAST.4
       LT1=LINBYT.SHR.12
       LT2=LINBYT.SHR.8
       LT2=LT2.AND.LT6
       LT3=LINBYT.SHR.4
       LT3=LT3.AND.LT6
       LT4=LINBYT.AND.LT6
A      %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       TIME(*)=((TIME(*)+T1)*60+T2*10+T3)*60+T4*10
      *CALL GETBYT
       LT1=LINBYT.SHR.12
A      %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       TIME(*)=(TIME(*)+T1)*10
      *IF (DEBUG.LT.1) GO TO 110
A      DISPLH "FMT#2",0:
      *IF(DEBUG.LT.2)GO TO 110
A      DISPLH "TIME:",16,TIME,TIME:
 110  *GO TO 500
 200  *CONTINUE
C      IF WE GET HERE SOMETHING WIERD IS GOING ON, SINCE IT AINT A VALID
C      FORMAT. PRINT OUT A LITTLE MESSAGE AND SEE WHAT HAPPENS.
A      DISPLH "ARGTIME",2:
 500  *CONTINUE
      *IF(DEBUG.LT.1)GO TO 510
A      DISPLH "ECNVTIM:",0:
 510  *RETURN
      *END
      *SUBROUTINE FNGRID
C       PURPOSE:ONCE A POWER MAXIMUM HAS BEEN FOUND FOR EACH FREQUENCY
C       ON THE COARSE GRID REFINEMENTS OF THE MAX ARE PERFORMED ON A
C       SQUARE GRID. THE SPACING ON THE SQUARE GRID IS DKX/6.
C       THE SEARCH STARTS IN THFEAST DIRECTION AND PROCEEDS
C       CLOCKWISE. THE DIRECTION OF THE SEARCH IS CHANGED WHEN A
C       MAX IS FOUND IN THA DIRECTION.
C       DECLARATIONS:
      *PE INTEGER INBUF(*,640),CNTRL(*,6),NCHAN(*),PINT1(*),OFFSET(*),   -
      1           LOCATE(*),NPTS(*),COUNT2(*),COUNT3(*),LOC2D(*,25),     -
      2           LOC3D(*,25),TWTIME(*),ADJF(*)
      *PE REAL POWER(*,25),FMAX(*,25),FKX(*,25),FKY(*,25),RINBUF(*,640), -
      1        X(*,25),Y(*,25),FFT(*,612),KERNEL(*,25),                  -
      2        XCOORD(*),YCOORD(*),PREAL1(*),      COSK(*),SINK(*),      -
      4        COSDK(*),SINDK(*),BEAMER(*),FPMAX(*),KXMAX(*),            -
      5        KYMAX(*),DELX(*),DELY(*),KXSEP(*),KYSEP(*),KSEP(*),       -
      6        VEL(*),AZ(*),SIGNAL(*),FSTAT(*),SUMSQ(*)    ,             -
      7        TEST(*),K(*),CHANAV(*),TPOWER(*),FREQ(*),TDKX(*)
      *PE REAL ADKX(4),ADKY(4),YPOINT(50),YMAX(50),DX(500),DY(500)
      *PE REAL BEAM(*),TPOW(*),DELTAK(*),NRPOWR(*,25),NIPOWR(*,25),P
      *PE REAL PREAL2(*),RPOWER(*,25),IPOWER(*,25),RTPOW(*),ITPOW(*)
      *PE INTEGER MAX
      *CU INTEGER LOFREQ,HIFREQ,DEBUG,SM,T1,T2,ARRAY,PAGE,I,N,MNCHAN,    -
      1           MNPTS,NPOINT,SWITCH,NFREQ,IGO,LINE,LINES,INDEX,IP,     -
      2           TWIN,SAM,IFREQ,J,NFREQ1,REFINE,IND,YTOP        ,       -
      3            YPMI,SIGN,NTIMES,LINEP1
      *PE REAL DELTX(3000),DELTY(3000),DIST
      *PE REAL DELTAX,DELTAY,KX,KY
      *CU REAL DKX,LOWER,UPPER,LINEP,HDKX,BORDER,TWOH
      *CU REAL DELTAF,RADIUS,ANGLE
      *CU LOGICAL MODE3,NMODE
      *EXTERNAL MAX,GRID,REALE,IMG
      *COMMON/MAINFK/INBUF,CNTRL,NCHAN,PINT1,OFFSET,LOCATE,NPTS,COUNT2,  -
      1              COUNT3,LOC2D,LOC3D,POWER,FMAX,FKX,FKY,X,Y,KERNEL,   -
      2                    XCOORD,YCOORD,PREAL1      ,COSK,SINK,BEAM,    -
      3              TPOW,DELTAK,RPOWER,IPOWER,              COSDK,SINDK, -
      4              BEAMER,FPMAX,KXMAX,KYMAX,DELX,DELY,KXSEP,KYSEP,KSEP,-
      5              TWTIME,TPOWER,VEL,AZ,SIGNAL,FSTAT,SUMSQ,TEST,K,     -
      6              CHANAV,FREQ,ADJF,DX,DY,P,YPOINT,YMAX,ADKX,ADKY,     -
      7               KX,KY
      *EQUIVALENCE(INBUF(1,1),RINBUF(1,1)),(INBUF(1,28),FFT(1,1))
      *EQUIVALENCE (1,LOFREQ),(2,HIFREQ),(3,DEBUG),(4,SM),(5,T1),(6,T2), -
      1            (7,ARRAY),(8,PAGE),(9,I),(10,N),         (12,MNCHAN), -
      2            (13,MNPTS),(14,NPOINT),(15,SWITCH),(16,IGO),          -
      3                              (17,INDEX),(18,IP),(19,DKX),        -
      4            (20,LOWER),(21,UPPER),(22,LINE),(23,LINES),           -
      5                  (24,HDKX),(25,BORDER),(26,TWOH),                -
      6            (27,DELTAF),(28,RADIUS),           (29,SIGN),         -
      7            (30,MODE3),(31,NMODE),(32,TWIN),(33,ANGLE),(34,SAM),  -
                                         **
      8            (35,NFREQ),(36,IFREQ),(37,J),(38,NFREQ1),(39,REFINE)  -
```

```
     9             ,(40,NTIMES),(41,IND),(42,YTOP),(43,YPMI),(44,LINEP), -
     0             (45,LINEP1)
     *DISK AREA CONPRM(1),STCORD(1),FKIN(81)
      TDKX(*)=DKX
     *DO 500 NTIMES=1,REFINE
      TDKX(*)=TDKX(*)/6.0
      PREAL1(*)=TDKX(*)
      ADKX(1)=PREAL1(1)
      ADKX(2)=0.0
      ADKX(4)=0.0
      ADKY(1)=0.0
      ADKY(3)=0.0
      ADKY(4)=PREAL1(1)
      PREAL1(*)=-PREAL1(*)
      ADKX(3)=PREAL1(1)
      ADKY(2)=PREAL1(1)
     *DO 475 I=1,4
      MODE=ON
      DELX(*)=ADKX(I)
      DELY(*)=ADKY(I)
      RTPOW(*)=0.0
      ITPOW(*)=0.0
      MBIT1=MODE
     *DO 250 N=1,MNCHAN
      MODE=(N.LE.NCHAN(*))
      DELTAK(*)=+6.283185307*(DELX(*)*X(*,N)+DELY(*)*X(*,N))
      COSDK(*)=COS(DELTAK(*))
      SINDK(*)=SIN(DELTAK(*))
      NRPOWR(*,N)=RPOWER(*,N)*COSDK(*)-IPOWER(*,N)*SINDK(*)
      NIPOWR(*,N)=RPOWER(*,N)*SINDK(*)+IPOWER(*,N)*COSDK(*)
      RTPOW(*)=RTPOW(*)+NRPOWR(*,N)
      ITPOW(*)=ITPOW(*)+NIPOWR(*,N)
 250*CONTINUE
      MODE=MBIT1
      TPOW(*)=RTPOW(*)**2+ITPOW(*)**2
 400*CONTINUE
     *IF(DEBUG.LT.2)GO TO 405
A     DISPLH "FINE",0;
      PINT1(1)=I
A     DISPLH "I",16,PINT1,PINT1;
A     DISPLF "DELX",16,DELX,DELX;
A     DISPLF "DELY",16,DELY,DELY;
A     DISPLF "RTPOW",16,RTPOW,RTPOW+3;
A     DISPLF "ITPOW",16,ITPOW,ITPOW+3;
A     DISPLF "TPOW",16,TPOW,TPOW+3;
A     DISPLF "FPMAX",16,FPMAX,FPMAX+3;
 405*CONTINUE
      MODE=MODE.AND.(TPOW(*).GT.FPMAX(*))
     *IF(.NOTANY.(MODE))GO TO 475
      MBIT1=MODE
     *DO 410 J=1,MNCHAN
      RPOWER(*,J)=NRPOWR(*,J)
      IPOWER(*,J)=NIPOWR(*,J)
```

```
  410*CONTINUE
       MODE=MBIT1
       KXMAX(*)=KXMAX(*)+DELX(*)
       KYMAX(*)=KYMAX(*)+DELY(*)
       FPMAX(*)=TPOW(*)
       TPOW(*)=0.
       RTPOW(*)=0.0
       ITPOW(*)=0.0
      *DO 450 J=1,MNCHAN
       MODE=MODE.AND.(J.LE.NCHAN(*))
       DELTAK(*)=+6.2831853070*(DELX(*)*X(*,J)+DELY(*)*Y(*,J))
       COSDK(*)=COS(DELTAK(*))
       SINDK(*)=SIN(DELTAK(*))
       NRPOWR(*,J)=RPOWER(*,J)*COSDK(*)-IPOWER(*,J)*SINDK(*)
       NIPOWR(*,J)=RPOWER(*,J)*SINDK(*)+IPOWER(*,J)*COSDK(*)
       RTPOW(*)=RTPOW(*)+NRPOWR(*,J)
       ITPOW(*)=ITPOW(*)+NIPOWR(*,J)
  450*CONTINUE
       MODE=MBIT1
       TPOW(*)=RTPOW(*)**2+ITPOW(*)**2
      *GO TO 400
  475*CONTINUE
       MODE=ON
  500*CONTINUE
      *IF(DEBUG.LT.2)GO TO 510
A      DISPLF "EFNGRID",0;
 510  *CONTINUE
      *RETURN
      *F
      *SUBROUTINE GETBYT
C      GETBYT IS SUPPLIED A POINTER TO THE BYTE WANTED VIA INPTB. IT
C      FIGURES OUT IF THE BYTE IS IN ADB OR CORE OR ON DISK. IT GETS THE
C      DEMANDED BYTE AND LEAVES IT RIGHT JUSTIFIED IN "INBYT". ALL
C      POINTER START AT ZERO RATHER THAN ONE BECAUSE OF THE SHIFTING,
C      ETC. DONE.
      *PE INTEGER CNTRL(*,6),OUTBUF(*,64,6),PINT1(*),INBUF(*,128),        -
      1          TIME(*),OLDTIM(*),                                        -
      1                   SAVBCT,SAVPTW,OUPAGE(6),TSTEPS(6),SCANS,         -
      2           OUPTWA(6),      OTIMEA(6),ORGADB,INBUF1(8192)
      *CU INTEGER ADBBUF(8),ARRAY,INPTB,INPTW,SAVADB,ADBOUT(6),OUPTW,     -
      1           BYTS,WORDS,T1,T2,T3,T4,T5,T6,      IT,PRTIAL,ADDRS,      -
      2           WORD,     BYTCNT(6),ADBWRD,INBYT,OUBYT,ORGCOR,PAGE,     -
      3           DEBUG,BCT,ADB,ENDADB
      *CU LOGICAL LADBBU(8),LARRAY,LINPTB,LINPTW,LSAVAD,LADBOU(6),LOUPTW,-
      1           LBYTS,LWORDS,LT1,LT2,LT3,LT4,LT5,LT6,LOUBYT,LIT,LPRTIA,-
      2           LADDRS,LWORD,LINBYT,LBYTCN(6),LADBWR,LORGCO,LPAGE        -
      3           ,LDEBUG,LBCT,LADB,LENDAD
      *EXTERNAL RDPRM,PUTBYT,CNVTIM
      *COMMON/MAIN/CNTRL,OUTBUF,INBUF,PINT1,TIME,OLDTIM,SAVBCT,SAVPTW,    -
      1      TSTEPS,SCANS,OUPTWA,OUPAGE,OTIMEA,ORGADB
      *EQUIVALENCE(1,ADBBUF(1),LADBBU(1)),(9,ARRAY,LARRAY),               -
      1           (10,INPTB,LINPTB),                                       -
      1           (11,INPTW,LINPTW),(12,SAVADB,LSAVAD),                    -
      1           (13,ADBOUT(6),LADBOU(6))                                 -
      2           ,(19,OUPTW ,LOUPTW),(20,BYTS,LBYTS),(21,WORDS,LWORDS), -
      3           (22,T1,LT1),(23,T2,LT2),(24,T3,LT3),(25,T4,LT4),         -
      4           (26,T5,LT5),(27,T6,LT6),(28,OUBYT,LOUBYT),(29,IT,LIT), -
      5           (30,PRTIAL,LPRTIA),(31,ADDRS,LADDRS),(32,WORD,LWORD),   -
      6           (33,INBYT,LINBYT),(34,BYTCNT(1),LBYTCN(1)),              -
      6           (40,ADBWRD,LADBWR)                                       -
      7                                               ,(43,ORGCOR,LORGCO),-
      8           (44,PAGE,LPAGE),(45,DEBUG,LDEBUG),(46,BCT,LBCT),        -
      9           (47,ADB,LADB),(48,ENDADB,LENDAD)
      *EQUIVALENCE (INBUF(1,1),INBUF1(1))
      *DISK AREA OUPUT1(20),OUPUT2(20),OUPUT3(20),OUPUT4(20),OUPUT5(20), -
      1          OUPUT6(20),INPUT(50)
C      *IF(DEBUG.LT.1)GO TO 5
A      DISPLH "GETBYT",0;
      *IF(DEBUG.LT.3)GO TO 5
A      DISPLH "GETBYT1",2;
    5 *CONTINUE
C      FIRST MAKE WORD POINTER=BYTE POINTER/4.
       LINPTW=LINPTB.SHR.2
A      % DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      *IF(INPTW.LT.ENDADB)GO TO 30
C      IF WE GET HERE ADB MUST BE REFILLED.
      *IF(INPTW.LT.ORGCOR+8192)GO TO 20
C      IF WE GET HERE CORE MUST BE REFILLED.
C      PAGE WANTED IS BYTE POINTER/4096 + 1.
       LPAGE=LINPTB.SHR.12
```

```
A        %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
         PAGE=PAGE+1
        *READ(64,INBUF(1,1),INPUT(PAGE),8)
         LORGCO=LINPTB.SHR.12
         LORGCO=LORGCO.SHL.10
A        %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
C        ORGCOR NOW HAS THE WORD ADDRESS OF THE FIRST BYTE IN INBUF.
        *WAIT 64
C        A BIT OF DEBUG OUTPUT NOW
        *IF(DEBUG.LT.1)GO TO 20
A        DISPLH "REFILLC:",0;
        *IF(DEBUG.LT.3)GO TO 20
A        DISPLH "INBUF:",18,INBUF,INBUF+64*4-1;
   20   *CONTINUE
C        AT THIS POINT WE REFILL ADBBUF.
         LT6=LINPTB.SHR.5
         LT6=LT6.SHL.3
A        %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
         ORGADB=T6
         ENDADB=T6+8
C        ORGADB NOW HAS THE WORD ADDRESS OF THE FIRST BYTE IN ADB.ENDADB
C        HAS THE WORD ADDRESS OF THE BYTE AFTER THE LAST BYTE IN ADB.
         T5=T6-ORGCOR+1
C        THATS THE ADDRESS WITHIN INBUF WE WANT TO REFILL ADBBUF FROM.
        *TRANSFER(8) ADBBUF(1)=INBUF1(T5)
        *IF(DEBUG.LT.1) GO TO 30
A        DISPLH "REFILLA:",2;
   30   *CONTINUE
C        WHEN WE GET HERE THE BYTE WANTED IS IN ADBBUF. JUST GOTTA FIGURE
C        OUT WHERE.
         T6=ORGADB
         ADBWRD=INPTW-T6+1
A        % A DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
         LT6=OFF.TURN ON..LAST.2
         LT6=LINPTB.AND.LT6
         LT6=LT6.SHL.4
C         T6 IS USED TO DETERMINE THE AMOUNT TO ROTATE THE WORD IN ADB
C         TO RIGHT JUSTIFY THE BYTE WE WANT. NUMBERING THE BYTES 0,1,2,3,
C        THE BYTE WANTED IS (INPTB MOD 4). TO RIGHT JUSTIFY THAT BYTE WE
C        ROTATE LEFT BY ((INPTB MOD 4)+1)*16 OR (INPTB MOD 4)*16+16.T6
C        HAS JUSTED BEEN ASSIGNED (INPTB MOD 4)*16 SO WE HAVE ONLY TO ADD
C        16.
         LT5=OFF.TURN ON..LAST.16
A        %A DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
         LINBYT=LADBBU(ADBWRD).RTL.T6+16
         LINBYT=LINBYT.AND.LT5
        *IF(DEBUG.LT.1)GO TO 40
A        DISPLH "EGETBYT:",0;
        *IF(DEBUG.LT.3)GO TO 40
A        DISPLH "GETBYT:",2;
   40   *CONTINUE
         INPTB=INPTB+1
        *RETURN
        *END
```

```
     *SUBROUTINE GRID
C     DECLARATIONS:
     *PE INTEGER INBUF(*,640),CNTRL(*,6),NCHAN(*),PINT1(*),OFFSET(*),    -
     1           LOCATE(*),NPTS(*),COUNT2(*),COUNT3(*),LOC2D(*,25),      -
     2           LOC3D(*,25),TWTIME(*),ADJF(*)
     *PE REAL POWER(*,25),FMAX(*,25),FKX(*,25),FKY(*,25),RINBUF(*,640), -
     1        X(*,25),Y(*,25),FFT(*,612),KERNEL(*,25),                   -
     2        XCOORD(*),YCOORD(*),PREAL1(*),      COSK(*),SINK(*),       -
     4        COSDK(*),SINDK(*),BEAMER(*),FPMAX(*),KXMAX(*),             -
     5        KYMAX(*),DELX(*),DELY(*),KXSEP(*),KYSEP(*),KSEP(*),        -
     6        VEL(*),AZ(*),SIGNAL(*),FSTAT(*),SUMSQ(*)       ,           -
     7        TEST(*),K(*),CHANAV(*),TPOWER(*),FREQ(*)
     *PE REAL ADKX(4),ADKY(4),YPOINT(50),YMAX(50),DX(500),DY(500)
     *PE REAL BEAM(*),TPOW(*),DELTAK(*),P
     *PE REAL PREAL2(*),RPOWER(*,25),IPOWER(*,25),RTPOW(*),ITPOW(*)
     *PE INTEGER MAX
     *CU INTEGER LOFREQ,HIFREQ,DEBUG,SM,T1,T2,ARRAY,PAGE,I,N,MNCHAN,     -
     1           MNPTS,NPOINT,SWITCH,NFREQ,IGO,LINE,LINES,INDEX,IP,      -
     2           TWIN,SAM,IFREQ,J,NFREQ1,REFINE,IND,YTOP           ,     -
     3            YPM1,NTIMES,LINEP1
     *PE REAL DELTX(3000),DELTY(3000),DIST
     *PE REAL DELTAX,DELTAY,KX,KY
     *CU REAL DKX,LOWER,UPPER,LINEP,HDKX,BORDER,TWOH,SIGN
     *CU REAL DELTAF,RADIUS,ANGLE
     *CU LOGICAL MODE3,NMODE
     *EXTERNAL MAX,FNGRID,REALE,IMG
     *COMMON/MAINFK/INBUF,CNTRL,NCHAN,PINT1,OFFSET,LOCATE,NPTS,COUNT2,   -
     1              COUNT3,LOC2D,LOC3D,POWER,FMAX,FKX,FKY,X,Y,KERNEL,    -
     2                     XCOORD,YCOORD,PREAL1       ,COSK,SINK,BEAM,   -
     3              TPOW,DELTAK,RPOWER,IPOWER,             COSDK,SINDK,  -
     4              BEAMER,FPMAX,KXMAX,KYMAX,DELX,DELY,KXSEP,KYSEP,KSEP,-
     5              TWTIME,TPOWER,VEL,AZ,SIGNAL,FSTAT,SUMSQ,TEST,K,      -
     6              CHANAV,FREQ,ADJF,DX,DY,P,YPOINT,YMAX,ADKX,ADKY,      -
     7               KX,KY,DELTX,DELTY
     *EQUIVALENCE(INBUF(1,1),RINBUF(1,1)),(INBUF(1,28),FFT(1,1))
     *EQUIVALENCE (1,LOFREQ),(2,HIFREQ),(3,DEBUG),(4,SM),(5,T1),(6,T2),  -
     1            (7,ARRAY),(8,PAGE),(9,I),(10,N),         (12,MNCHAN),  -
     2            (13,MNPTS),(14,NPOINT),(15,SWITCH),(16,IGO),           -
     3                                 (17,INDEX),(18,IP),(19,DKX),      -
     4            (20,LOWER),(21,UPPER),(22,LINE),(23,LINES),            -
     5                        (24,HDKX),(25,BORDER),(26,TWOH),           -
     6            (27,DELTAF),(28,RADIUS),          (29,SIGN),           -
     7            (30,MODE3),(31,NMODE),(32,TWIN),(33,ANGLE),(34,SAM),   -
                                                 **
     8            (35,NFREQ),(36,IFREQ),(37,J),(38,NFREQ1),(39,REFINE)   -
     9            ,(40,NTIMES),(41,IND),(42,YTOP),(43,YPM1),(44,LINEP),  -
     0             (45,LINEP1)
     *DISK AREA CONPRM(1),STCORD(1),FKIN(81)
     *IF(SWITCH.EQ.2)GO TO 500
      PREAL1(*)=DKX*(SQRT(2.0)/2.0)
      BORDER=PREAL1(1)
      PREAL1(*)=FLOAT(SAM)/FLOAT(TWIN)
      DELTAF=PREAL1(1)
      PREAL1(*)=ANGLE
     *IF(.ANY.((ANGLE.GT.0.0)))GO TO 20
      PREAL1(*)=(1.0/LOWER)*FLOAT(IFREQ)*DELTAF+BORDER
      RADIUS=PREAL1(1)
      KY=0.0
      KX=0.0
     *GO TO 30
   20 PREAL1(*)=(1.0/LOWER-1.0/UPPER)*0.5*FLOAT(IFREQ)*DELTAF+BORDER
      RADIUS=PREAL1(1)
      PREAL1(*)=RADIUS*SIN(ANGLE)
      KX=PREAL1(1)
      PREAL1(*)=RADIUS*COS(ANGLE)
      KY=PREAL1(1)
   30 PREAL1(*)=2.0*SQRT(2.0)*DKX*.75
      DELTAY=PREAL1(1)
      PREAL1(*)=(DKX*SQRT(6.0)/2.0)/2.0
      DELTAX=PREAL1(1)
C CHANGE IN X IS HALF THE BASE TRIANGLE,CHANGE IN Y IS TWICE
C THE HEIGHT OF BASE TRIANGLE.
      PINT1(*)=IFIX(RADIUS/DELTAX)-1
      LINE=PINT1(1)
      PREAL1(*)=FLOAT(LINE)*DELTAX
      P=PREAL1(1)
      PINT1(*)=2*LINE+1
      LINES=PINT1(1)
      IGO=1
      PINT1(*)=(LINE/2)*2
      T1=PINT1(1)
     *IF(LINE.EQ.T1) IGO=0
     *DO 100 INDEX=1,LINE
      PREAL1(*)=SQRT(RADIUS**2-(P-FLOAT( INDEX-1)*DELTAX)**2)
      DIST=PREAL1(1)
     *IF(IGO.EQ.0)GO TO 50
      PINT1(*)=IFIX((DIST-DELTAY/2.0)/DELTAY)
      YTOP=PINT1(1)
      PREAL1(*)=2.0*FLOAT(YTOP)+2.0
      YPOINT(INDEX)=PREAL1(1)
      PREAL1(*)=DELTAY/2.0+DELTAY*FLOAT(YTOP)
      YMAX(INDEX)=PREAL1(1)
      IGO=0
     *GO TO 70
   50 *CONTINUE
      PINT1(*)=IFIX(DIST/DELTAY)
      YTOP=PINT1(1)
      PREAL1(*)=2.0*FLOAT(YTOP)+1.0
      YPOINT(INDEX)=PREAL1(1)
      PREAL1(*)=FLOAT(YTOP)*DELTAY
      YMAX(INDEX)=PREAL1(1)
      IGO=1
```

```
 70  IND=LINES-INDEX+1
     YPOINT(IND)=YPOINT(INDEX)
     YMAX(IND)=YMAX(INDEX)
100 *CONTINUE
     PREAL1(*)=KX-P
     DELTX(1)=PREAL1(1)
     PREAL1(*)=KY-YMAX(1)
     DELTY(1)=PREAL1(1)
     DX(1)=DELTX(1)
     DY(1)=DELTY(1)
     LINEP1=LINE+1
     PINT1(*)=IFIX(RADIUS/DELTAY)
     YTOP=PINT1(1)
     PREAL1(*)=FLOAT(2*YTOP+1)
     YPOINT(LINEP1)=PREAL1(1)
     PREAL1(*)=FLOAT(YTOP)*DELTAY
     YMAX(LINEP1)=PREAL1(1)
     I=1
     SIGN=1.
     INDEX=0
150  INDEX=INDEX+1
     PINT1(*)=IFIX(YPOINT(INDEX))-1
     YPMI=PINT1(1)
    *DO 160 IND=1,YPMI
     I=I+1
     DELTX(I)=0.0
     PREAL1(*)=SIGN*DELTAY
     DELTY(I)=PREAL1(1)
     PREAL1(*)=DX(I-1)+DELTX(I)
     DX(I)=PREAL1(1)
     PREAL1(*)=DY(I-1)+DELTY(I)
     DY(I)=PREAL1(1)
160 *CONTINUE
    *IF(INDEX.EQ.LINES)GO TO 200
     I=I+1
     DELTX(I)=DELTAX
     PREAL1(*)=(YMAX(INDEX+1)-YMAX(INDEX))*SIGN
     DELTY(I)=PREAL1(1)
     PREAL1(*)=DX(I-1)+DELTX(I)
     DX(I)=PREAL1(1)
     PREAL1(*)=DY(I-1)+DELTY(I)
     DY(I)=PREAL1(1)
     PREAL1(*)=-SIGN
     SIGN=PREAL1(1)
    *GO TO 150
200 *CONTINUE
     NPTS(*)=I
    *GO TO 515
500 *CONTINUE
    *DO 510 T1=1,64
     T2=LOCATE(T1)
     KXMAX(T1)=DX(T2)
     KYMAX(T1)=DY(T2)
 510 *CONTINUE
 515 *CONTINUE
 520 *CONTINUE
     *RETURN
     *END
```

```
      *SUBROUTINE GTDATA
C      GTDATA GETS ONE 16-BIT BYT FROM THE INPUT FILE. THE BYTES ARE
C      DELIVERED SEQUENTIALLY STARTING AT THE BEGINNING OF THE DISK AREA
C      INDM2. THERE IS A CORE BUFFER (BUUF1(-)) AND AN 8 WORD ADB BUFFER
C      SO ALL MEMORY ACCESSES ARE DONE VIA THE BIN INSTRUCTION. THE BYTE
C      ENDS UP RIGHT JUSTIFIED IN "INBYT".
      *PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),    -
      1           PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
      *PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
      1       PREAL2(*),ALLMSQ(*),TVARFT(*)
      *PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
      1           CHGOOD(80),SITEGD(80),SITES(80)
      *PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
      *CU INTEGEP ADBBUF(8),COREPT,      BYTE,ADBWRD,ARRAY,DEBUG,TWSZ,    -
      1           OVLAP,NCHAN,NSITE,NROWS,DIFFR,DIFFW,NEW,OLD,GAP,TSCANS,-
      2          INDEX1,INDEX2,INDEX3,INDEX4,T1,T2,T3,T4,T5,T6,CH,IPAGE, -
      3          OFFSET,INBYT,NGDCH,TWSZR,NGDST,NGDR,F,BF3PE,NGT,OPAGE,T7
      *CU LOGICAL LADBBU(8),LCOREP,LAST16,LBYTE,LADBWR,LARRAY,LDEBUG,    -
      1           LTWSZ,LOVLAP,LNCHAN,LNSITE,LNROWS,LDIFFR,LDIFFW,LNEW,  -
      2           LOLD,LGAP,LTSCAN ,LT1,LT2,LT3,LT4,LT5,LT6,LCH,LOFFSE,  -
      3           LINBYT,LF,LNGDCH,LTWSZR,LNGDST,LNGDR,LNGT,LT7
      *EXTERNAL C16T64,C64T32,ROWSUM,RUNFFT,C32T64
      *COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,   -
      1      TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,     -
      2      ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
      *EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),      -
      1             ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),          -
      2           (BUFF3(1,1),IBUFF3(1,1))
      *EQUIVALENCE (1,ADBBUF(1),LADPBU(1)),(9,COREPT,LCOREP),(10,BYTE,   -
      1           LBYTE),(11,ADBWRD,LADBWR),(12,ARRAY,LARRAY),(13,TWSZ,  -
      2           LTWSZ),(14,OVLAP,LOVLAP),(15,NCHAN,LNCHAN),(16,NSITE,  -
      3           LNSITE),(17,NROWS,LNROWS),(18,DIFFR,LDIFFR),(19,DIFFW, -
      4           LDIFFW),(20,NEW,LNEW),(21,OLD,LOLD),(22,GAP,LGAP),(23, -
      5           TSCANS,LTSCAN),(24,INDEX1),(25,INDEX2),(26,INDEX3),(27,-
      6           INDEX4),(28,T1,LT1),(29,T2,LT2),(30,T3,LT3),(31,T4,    -
      7           LT4),(32,T5,LT5),(33,T6,LT6),(34,CH,LCH),(35,OFFSET,   -
      8           LOFFSE),(36,INBYT,LINBYT),(37,F,LF),(38,NGDCH,LNGDCH), -
      9           (39,TWSZR,LTWSZR),(40,NGDST,LNGDST),(41,NGDR,LNGDR),   -
      0           (42,BF3PE),(43,NGT,LNGT),(44,LAST16),(45,DEBUG,LDEBUG),-
      1           (46,OPAGE),(47,T7,LT7),(48,IPAGE)
      *DISK AREA INDM2(20),OUTDM2(40),CONPRM(1)
      *IF(DEBUG.LT.1)GO TO 10
A      DISPLH "BGTDATA",0:
      *IF(DEBUG.LT.4)GO TO 10
A      DISPLH ,2:
   10 *CONTINUE
       BYTE=BYTE+1
      *IF(BYTE.LT.5)GO TO 300
C      HAVE TO GO TO NEXT ADB WORD.
       ADBWRD=ADBWRD+1
      *IF(ADBWRD.LT.9)GO TO 200
```

```
C      HAVE TO REFILL ADB BUFFER.
       COREPT=COREPT+8
      *IF(COREPT.LT.4097)GO TO 100
C      HAVE TO REFILL BUFF1.
      *READ(64,IBUFF1(1),INDM2(IPAGE),4)
      *WAIT 64
       IPAGE=IPAGE+4
       COREPT=1
      *IF(DEBUG.LT.1)GO TO 20
A       DISPLH "REFILLC",0:
      *IF(DEBUG.LT.2)GO TO 20
A       DISPLH,18,IBUFF1,IBUFF1+255:
   20 *CONTINUE
  100 *CONTINUE
C      CORE IS OKAY. HAVE TO REFILL ADB BUFFER.
      *TRANSFER(8) ADBBUF(1)=IBUFF1(COREPT)
       ADBWRD=1
      *IF(DEBUG.LT.1)GO TO 110
A       DISPLH "REFILLA",0:
      *IF(DEBUG.LT.2)GO TO 110
A       DISPLH ,2:
  110 *CONTINUE
  200 *CONTINUE
C      NEW ADB WORD.
       BYTE=1
      *IF(DEBUG.LT.1)GO TO 210
A      DISPLH "NEW WORD",0:
      *IF(DEBUG.LT.2)GO TO 210
A      DISPLH ,2:
  210 *CONTINUE
  300 *CONTINUE
       LADBBU(ADBWRD)=LADBBU(ADBWRD).RTL.16
       LINBYT=LADBBU(ADBWRD).AND.LAST16
      *IF(DEBUG.LT.1)GO TO 310
A       DISPLH "EGETBYT",0:
      *IF(DEBUG.LT.4)GO TO 310
A       DISPLH ,2:
  310 *CONTINUE
      *RETURN
      *END
```

```
;  <KERR>CFD.IMG;4   THU 18-JUL-74 8:41AM                                  PAGE 1

      *SUBROUTINE IMG(IN,OUT)
      *PE REAL IN(*),OUT(*)
      *CU INTEGER DEBUG
      *EQUIVALENCE (3,DEBUG)
      *IF(DEBUG.LT.5)GO TO 10
A      DISPLH "IMG",0;
      *IF(DEBUG.LT.5)GO 10
A      LDL(0) $D49;
A      LDA IN(0);
A      DISPLH "IN",32;
   10 *CONTINUE
A      LDL(0) $D49;
A      LDA IN(0);
A      LDR $A;                     % SAVE IT.
A      SHAL =9;
A      SHAR =57;           % ISOLATE THE EXPONENT.
A      SBM =40:16;         % SUBTRACT THE 32 BIT OFFSET.
A      ADM =4000:16;       % ADD THE 64 BIT OFFSET.
A      SHAL =48;           % PUT IT IN THE 64-BIT EXPONENT FIELD.
A      LDS $A;             % SAVE IT FOR NOW.
A      LDA $R;             % NOW FOR THE SIGN.
A      SHAR =55;           % START ISOLATING THE SIGN.
A      SHAL =63;           % COMPLETE ISOLATION AND PUT IT IN RIGHT SPOT.
A      OR $S;              % NOW WE HAVE EXPOONENT AND SIGN.
A      LDS $A;             % SAVE IT.
A      LDA $R;             % RESTORE ORIGINAL TO DO MANTISSA.
A      SHAL =16;           % START ISOLATING.
A      SHAR =40;           % REMOVE OUTER MANTISSA.
A      SHAL =24;           % COMPLETE ISOLATION.
A      OR $S;              % PUT THE THREE PARTS TOGETHER.
A      LDL(0) $D50;
A      STA OUT(0);
      *IF(DEBUG.LT.5)GO TO 110
A      LDL(0) $D50;
A      LDA OUT(0);
A      DISPLH "OUT",32;
  110 *CONTINUE
      *RETURN
      *END
```

```
;  <KERR>CFD.MAX;10   THU 18-JUL-74 8:45AM                            PAGE 1

      *FUNCTION MAX(I)
      *PE INTEGER I(*)
      *PE INTEGER MAX
      *CU INTEGER DEBUG
      *EQUIVALENCE (3,DEBUG)
      *IF(DEBUG.LT.5)GO TO 5
A      DISPLH "B MAX",0;
    5 *CONTINUE
A      LDL(0) $D49;
A      LDA I(0);
A      LIT(0)=1,6,1;
A      LIT(1) =1;
A      CLC(3);
A      COMP(3);
A      SETC(2) E;
A      LDEE1 $C3;
A      RTL $A,0(1);
A      IMG $R;
A      SETE -I.AND.E;
A      SETE1 E.OR.E;
A      LDA $R;
A      CSHL(1) 1;
A      TXEFM(0) ,MAXL(X)P;
A      MAXL(X)P:LDEE1 $C2;
A      STA MFUNVAL;
      *IF(DEBUG.LT.5)GO TO 10
A      LDL(0) $D49;
A      LDA I(0);
A      DISPLH "I",32;
A      DISPLH "MAX",16,MFUNVAL,MFUNVAL+63;
   10 *CONTINUE
      *RETURN
      *END
      *SUBROUTINE OUTPUT
C      PURPOSE: COMPLETE CALCULATIONS ON THE POWER MAXIMUMS
C      BEFORE PRINTING OUTPUT.
C      VELOCITY AND AZIMUTH ARE CALCULATED AND SIGNAL TO NOISE
C      RATIO AND FISHER STATISTIC.
C      TWO AND THREE DIMENSIONAL MAXIMUM ARE OUTPUT IN SEPARATE
C      LISTS.
C      DECLARATIONS:
      *PE INTEGER INBUF(*,640),CNTRL(*,6),NCHAN(*),PINT1(*),OFFSET(*),
      1           LOCATE(*),NPTS(*),COUNT2(*),COUNT3(*),LOC2D(*,25),
      2           LOC3D(*,25),TWTIME(*),ADJF(*)
      *PE REAL POWER(*,25),FMAX(*,25),FKX(*,25),FKY(*,25),RINBUF(*,640),
      1        X(*,25),Y(*,25),FFT(*,612),KERNEL(*,25),
      2        XCOORD(*),YCOORD(*),PREAL1(*),PREAL2(*),COSK(*),SINK(*),
      4        COSDK(*),SINDK(*),BEAMER(*),FPMAX(*),KXMAX(*),
      5        KYMAX(*),DELX(*),DELY(*),KXSEP(*),KYSEP(*),KSEP(*),
      6        VEL(*),AZ(*),SIGNAL(*),FSTAT(*),SUMSQ(*)    ,
      7        TEST(*),K(*),CHANAV(*),TPOWER(*),FREQ(*)
      *PE REAL GROUP1(*),GROUP2(*),AFREQ(*,20),AK(*,20),AFSTAT(*,20),
      1        AAZ(*,20),AVEL(*,20),ACHANA(*,20),AFMAX(*,20),
      2        AGRUP1(*,20),AGRUP2(*,20),ASIGNA(*,20)
      *PE REAL ADKX(4),ADKY(4),YPOINT(50),YMAX(50),DX(50),DY(50)
      *PE REAL BEAM(*),TPOW(*),DELTAK(*),P,RPOWER(*,25),IPOWER(*,25)
      *PE INTEGER MAX
      *CU INTEGER LOFREQ,HIFREQ,DEBUG,SM,T1,T2,ARRAY,PAGE,I,N,MNCHAN,
      1           MNPTS,NPOINT,SWITCH,NFREQ,IGO,LINE,LINES,INDEX,IP,
      2           TWIN,ANGLE,SAM,IFREQ,J,NFREQ1,REFINE,IND,YTOP
      3            YPMI,SIGN,NTIMES,LINEP1
      *PE REAL DELTX(3000),DELTY(3000),DIST
      *PE REAL DELTAX,DELTAY,KX,KY
      *CU REAL DKX,LOWER,UPPER,LINEP,HDKX,BORDER,TWOH
      *CU REAL DELTAF,RADIUS
      *CU LOGICAL MODE3,NMODE
      *EXTERNAL MAX,FNGRID,GRID,REALE,IMG
      *COMMON/MAINFK/INBUF,CNTRL,NCHAN,PINT1,OFFSET,LOCATE,NPTS,COUNT2,
      1              COUNT3,LOC2D,LOC3D,POWER,FMAX,FKX,FKY,X,Y,KERNEL,
      2                    XCOORD,YCOORD,PREAL1      ,COSK,SINK,BEAM,
      3              TPOW,DELTAK,RPOWER,IPOWER,             COSDK,SINDK,
      4              BEAMER,FPMAX,KXMAX,KYMAX,DELX,DELY,KXSEP,KYSEP,KSEP,
      5              TWTIME,TPOWER,VEL,AZ,SIGNAL,FSTAT,SUMSQ,TEST,K,
      6              CHANAV,FREQ,ADJF,DX,DY,P,YPOINT,YMAX,ADKX,ADKY,
      7               KX,KY
      *EQUIVALENCE(INBUF(1,1),RINBUF(1,1)),(INBUF(1,28),FFT(1,1))
      *EQUIVALENCE (1,LOFREQ),(2,HIFREQ),(3,DEBUG),(4,SM),(5,T1),(6,T2),
      1            (7,ARRAY),(8,PAGE),(9,I),(10,N),          (12,MNCHAN),
      2            (13,MNPTS),(14,NPOINT),(15,SWITCH),(16,IGO),
      3                                (17,INDEX),(18,IP),(19,DKX),
      4            (20,LOWER),(21,UPPER),(22,LINE),(23,LINES),
      5                      (24,HDKX),(25,BORDER),(26,TWOH),
      6            (27,DELTAF),(28,RADIUS),            (29,SIGN),
      7            (30,MODE3),(31,NMODE),(32,TWIN),(33,ANGLE),(34,SAM),
      8            (35,NFREQ),(36,IFREQ),(37,J),(38,NFREQ1),(39,REFINE)
      9            ,(40,NTIMES),(41,IND),(42,YTOP),(43,YPMI),(44,LINEP),
      0             (45,LINEP1)
      *DISK AREA CONPRM(1),STCORD(1),FKIN(81)
      *IF(DEBUG.LT.1)GO TO 10
```

```
A      DISPLH"OUTPUT",0;
      *IF(DEBUG.LT.1)GO TO 10
A      DISPLH "COUNT2",16,COUNT2,COUNT2+10;
A      DISPLH "COUNT3",16,COUNT3,COUNT3+10;
A      DISPLH "LOC2D(1)",16,LOC2D,LOC2D+10;
A      DISPLH "LOC3D(1)",16,LOC3D,LOC3D+10;
    10*CONTINUE
C      COMPUTATIONS TO CALCULATE AZIMUTH OFF NORTH AND VELOCITY
C      IN KM PER SECOND.
       MODE=ON
       PINT1(*)=MAX(COUNT2(*))
       T1=PINT1(1)
      *DO 200 I=1,T1
       MODE=MODE.AND.(I.LE.COUNT2(*))
       MBIT2=MODE
       PINT1(*)=LOC2D(*,I)
       FMAX(*,PINT1(*))=FMAX(*,PINT1(*))/FLOAT(NCHAN(*)*TWIN)**2
       AZ(*)=ATAN(FKX(*,PINT1(*))/FKY(*,PINT1(*)))
       MBIT1=(FKY(*,PINT1(*)).LT.0.0)
       MODE=MODE.AND.MBIT1
       AZ(*)=AZ(*)+3.14159265
       MODE=MBIT2
       AZ(*)=AZ(*)*57.3
C      RADIANS TO DEGREES.
      *IF((AZ(*).LT.0.0))AZ(*)=AZ(*)+360.0
       K(*)=SQRT(FKX(*,PINT1(*))**2+FKY(*,PINT1(*))**2)
       VEL(*)=FLOAT(PINT1(*)-1)*DELTAF/K(*)
       CHANAV(*)=0.0
       OFFSET(*)=(PINT1(*)-LOFREQ)*NCHAN(*)
       MBIT1=MODE
      *DO 100 T2=1,MNCHAN
       MODE=MODE.AND.(T2.LE.NCHAN(*))
      *CALL REALE(FFT(*,OFFSET(*)+T2),PREAL1(*))
      *CALL IMG(FFT(*,OFFSET(*)+T2),PREAL2(*))
       CHANAV(*)=CHANAV(*)+PREAL1(*)**2+PREAL2(*)**2
   100*CONTINUE
       MODE=MBIT1
       CHANAV(*)=CHANAV(*)/FLOAT(NCHAN(*)*TWIN*TWIN)
       SIGNAL(*)=FMAX(*,PINT1(*))/(CHANAV(*)-FMAX(*,PINT1(*)))
       FSTAT(*)=(FLOAT(NCHAN(*))-1.0)*SIGNAL(*)
       GROUP1(*)=SQRT((FKX(*,PINT1(*))-FKX(*,PINT1(*)-1))**2+
      1                (FKY(*,PINT1(*))-FKY(*,PINT1(*)-1))**2)
       GROUP1(*)=DELTAF/GROUP1(*)
       GROUP2(*)=SQRT((FKX(*,PINT1(*))-FKX(*,PINT1(*)+1))**2+
      1                (FKY(*,PINT1(*))-FKY(*,PINT1(*)+1))**2)
       GROUP2(*)=DELTAF/GROUP2(*)
       AFREQ(*,I)=FLOAT(PINT1(*))
       AK(*,I)=K(*)
       AFSTAT(*,I)=FSTAT(*)
       AAZ(*,I)=AZ(*)
       AVEL(*,I)=VEL(*)
       ACHANA(*,I)=CHANAV(*)
       AFMAX(*,I)=FMAX(*,PINT1(*))
       AGRUP1(*,I)=GROUP1(*)
       AGRUP2(*,I)=GROUP2(*)
       ASIGNA(*,I)=SIGNAL(*)
      *IF(DEBUG.LT.1)GO TO 200
```

```
A       DISPLH "2-D MAX",0;
A       SETC(0) E;
A       DISPLH "MODE",1;
A       DISPLH "FREQ",16,PINT1,PINT1+63;
A       DISPLF "K",16,K,K+63;
A       DISPLF "FSTAT",16,FSTAT,FSTAT+63;
A       DISPLF "AZ",16,AZ,AZ+63;
A       DISPLF "VEL",16,VEL,VEL+63;
A       DISPLF "CHANAV",16,CHANAV,CHANAV+63;
A       DISPLF "SIGNAL",16,SIGNAL,SIGNAL+63;
        PREAL1(*)=FMAX(*,PINT1(*))
A       DISPLF "FMAX",16,PREAL1,PREAL1+63;
  200*CONTINUE
        MODE=ON
       *DO 250 I=1,64
        T1=NCHAN(I)
       *IF(T1.LT.1)GO TO 245
        T1=COUNT2(I)
       *IF(T1.LT.1)GO TO 245
        PINT1(1)=TWTIME(I)
A       DISPLH "TWTIME",16,PINT1,PINT1;
       *DO 240 J=1,T1
        PREAL1(1)=AFREQ(I,J)
        PREAL1(2)=AK(I,J)
        PREAL1(3)=AFSTAT(I,J)
        PREAL1(4)=AAZ(I,J)
        PREAL1(5)=AVEL(I,J)
        PREAL1(6)=ACHANA(I,J)
        PREAL1(7)=AFMAX(I,J)
        PREAL1(8)=AGRUP1(I,J)
        PREAL1(9)=AGRUP2(I,J)
        PREAL1(10)=ASIGNA(I,J)
A       SKIP ,E2D1;
A       B2D1:DATA
A       "FREQ              K                    FSTAT                AZ";
A       E2D1::;
A       DISPLS ,16,B2D1,E2D1-1;
A       DISPLF,16,PREAL1,PREAL1+3;
A       SKIP ,E2D2;
A       B2D2:DATA
A       "VEL              CHANAV              FMAX";
A       E2D2::;
A       DISPLS ,16,B2D2,E2D2-1;
A       DISPLF ,16,PREAL1+4,PREAL1+6;
A       SKIP ,E2D3;
A       B2D3:DATA
A       "GROUP1             GROUP2              SIGNAL/NOISE";
A       E2D3::;
A       DISPLS ,16,B2D3,E2D3-1;
A       DISPLF ,16,PREAL1+7,PREAL1+9;
 240 *CONTINUE
 245 *CONTINUE
 250 *CONTINUE
      PINT1(*)=MAX(COUNT3(*))
      T1=PINT1(1)
     *DO 400 I=1,T1
      MODE=MODE.AND.(I.LE.COUNT3(*))
      MBIT2=MODE
      PINT1(*)=LOC3D(*,I)
      FMAX(*,PINT1(*))=FMAX(*,PINT1(*))/FLOAT(NCHAN(*)*TWIN)**2
      AZ(*)=ATAN(FKX(*,PINT1(*))/FKY(*,PINT1(*)))
      MBIT1=(FKY(*,PINT1(*)).LT.0.0)
      MODE=MODE.AND.MBIT1
      AZ(*)=AZ(*)+3.14159265
      MODE=MBIT2
      AZ(*)=AZ(*)*57.3
C     RADIANS TO DEGREES.
     *IF((AZ(*).LT.0.0))AZ(*)=AZ(*)+360.0
      K(*)=SQRT(FKX(*,PINT1(*))**2+FKY(*,PINT1(*))**2)
      VEL(*)=FLOAT(PINT1(*)-1)*DELTAF/K(*)
      CHANAV(*)=0.0
      OFFSET(*)=(PINT1(*)-LOFREQ)*NCHAN(*)
      MBIT1=MODE
     *DO 300 T2=1,MNCHAN
      MODE=MODE.AND.(T2.LE.NCHAN(*))
     *CALL REALE(FFT(*,OFFSET(*)+T2),PREAL1(*))
     *CALL IMG(FFT(*,OFFSET(*)+T2),PREAL2(*))
      CHANAV(*)=CHANAV(*)+PREAL1(*)**2+PREAL2(*)**2
  300*CONTINUE
      MODE=MBIT1
      CHANAV(*)=CHANAV(*)/FLOAT(NCHAN(*)*TWIN*TWIN)
      SIGNAL(*)=FMAX(*,PINT1(*))/(CHANAV(*)-FMAX(*,PINT1(*)))
      FSTAT(*)=(FLOAT(NCHAN(*))-1.0)*SIGNAL(*)
      GROUP1(*)=SQRT((FKX(*,PINT1(*))-FKX(*,PINT1(*)-1))**2+
     1              (FKY(*,PINT1(*))-FKY(*,PINT1(*)-1))**2)
      GROUP1(*)=DELTAF/GROUP1(*)
      GROUP2(*)=SQRT((FKX(*,PINT1(*))-FKX(*,PINT1(*)+1))**2+
     1              (FKY(*,PINT1(*))-FKY(*,PINT1(*)+1))**2)
      GROUP2(*)=DELTAF/GROUP2(*)
      AFREQ(*,I)=FLOAT(PINT1(*))
      AK(*,I)=K(*)
      AFSTAT(*,I)=FSTAT(*)
      AAZ(*,I)=AZ(*)
      AVEL(*,I)=VEL(*)
      ACHANA(*,I)=CHANAV(*)
      AFMAX(*,I)=FMAX(*,PINT1(*))
      AGRUP1(*,I)=GROUP1(*)
      AGRUP2(*,I)=GROUP2(*)
      ASIGNA(*,I)=SIGNAL(*)
A     DISPLH "3-D MAX",0;
A     SETC(0) E;
A     DISPLH "MODE",1;
```

```
A      DISPLH "FREQ",16,PINT1,PINT1+63;
A      DISPLF "K",16,K,K+63;
A      DISPLF "FSTAT",16,FSTAT,FSTAT+63;
A      DISPLF "AZ",16,AZ,AZ+63;
A      DISPLF "VEL",16,VEL,VEL+63;
A      DISPLF "CHANAV",16,CHANAV,CHANAV+63;
A      DISPLF "SIGNAL",16,SIGNAL,SIGNAL+63;
       PREAL1(*)=FMAX(*,PINT1(*))
A      DISPLF "FMAX",16,PREAL1,PREAL1+63;
A      DISPLF "SIGNAL",16,SIGNAL,SIGNAL+63;
  400*CONTINUE
       MODE=ON
      *DO 450 I=1,64
       T1=NCHAN(I)
      *IF(T1.LT.1)GO TO 445
       T1=COUNT3(I)
      *IF(T1.LT.1)GO TO 445
       PINT1(1)=TWTIME(I)
A      DISPLH "TWTIME",16,PINT1,PINT1;
      *DO 440 J=1,T1
       PREAL1(1)=AFREQ(I,J)
       PREAL1(2)=AK(I,J)
       PREAL1(3)=AFSTAT(I,J)
       PREAL1(4)=AAZ(I,J)
       PREAL1(5)=AVEL(I,J)
       PREAL1(6)=ACHANA(I,J)
       PREAL1(7)=AFMAX(I,J)
       PREAL1(8)=AGRUP1(I,J)
       PREAL1(9)=AGRUP2(I,J)
       PREAL1(10)=ASIGNA(I,J)
A      DISPLS ,16,B2D1,E2D1-1;
A      DISPLF,16,PREAL1,PREAL1+3;
A      DISPLS ,16,B2D2,E2D2-1;
A      DISPLF ,16,PREAL1+4,PREAL1+6;
A      DISPLS ,16,B2D3,E2D3-1;
A      DISPLF ,16,PREAL1+7,PREAL1+9;
 440  *CONTINUE
 445  *CONTINUE
 450  *CONTINUE
      *RETURN
      *END
```

```
     *SUBROUTINE PUTBYT
C     PUTBYT TAKES THE RIGHT HAND 16 BITS OF "OUBYT" AND OUTPUTS THEM
C     TO ONE OF THE FILES OUPUT1-OUPUT5 DEPENDING ON THE VALUE OF THE
C     VARIABLE "ARRAY". THE BYTE GOES THRU A BUFFER IN ADB AND A BUFFER
C     IN CORE ON ITS WAY TO DISK.
C
     *PE INTEGER CNTRL(*,6),OUTBUF(*,64,6),PINT1(*),INBUF(*,128),          -
     1           TIME(*),OLDTIM(*),                                        -
     1                          SAVBCT,SAVPTW,OUPAGE(6),TSTEPS(6),SCANS,   -
     2            OUPTWA(6),      OTIMEA(6)                                -
     *CU INTEGER ADBBUF(8),ARRAY,INPTB,INPTW,SAVADB,ADBOUT(6),OUPTW,       -
     1           BYTS,WORDS,T1,T2,T3,T4,T5,T6,       IT,PRTIAL,ADDRS,      -
     2           WORD,      BYTCNT(6),ADBWRD,INBYT,OUBYT,ORGCOR,PAGE,      -
     3           DEBUG,BCT,ADB,ENDADB
     *CU LOGICAL LADBBU(8),LARRAY,LINPTB,LINPTW,LSAVAD,LADBOU(6),LOUPTW,-
     1           LBYTS,LWORDS,LT1,LT2,LT3,LT4,LT5,LT6,LOUBYT,LIT,LPRTIA,-
     2           LADDRS,LWORD,LINBYT,LBYTCN(6),LADBWR,LORGCO,LPAGE         -
     3           ,LDEBUG,LBCT,LADB,LENDAD
     *EXTERNAL RDPRM,GETBYT,CNVTIM
     *COMMON/MAIN/CNTRL,OUTBUF,INBUF,PINT1,TIME,OLDTIM,SAVBCT,SAVPTW,      -
     1      TSTEPS,SCANS,OUPTWA,OUPAGE,      OTIMEA
     *EQUIVALENCE(1,ADBBUF(1),LADBBU(1)),(9,ARRAY,LARRAY),                 -
     1           (10,INPTB,LINPTB),                                        -
     1           (11,INPTW,LINPTW),(12,SAVADB,LSAVAD),                     -
     1           (13,ADBOUT(1),LADBOU(1))                                  -
     2           ,(19,OUPTW ,LOUPTW),(20,BYTS,LBYTS),(21,WORDS,LWORDS),    -
     3           (22,T1,LT1),(23,T2,LT2),(24,T3,LT3),(25,T4,LT4),          -
     4           (26,T5,LT5),(27,T6,LT6),(28,OUBYT,LOUBYT),(29,IT,LIT),    -
     5           (30,PRTIAL,LPRTIA),(31,ADDRS,LADDRS),(32,WORD,LWORD),     -
     6           (33,INBYT,LINBYT),(34,BYTCNT(1),LBYTCN(1)),               -
     6           (40,ADBWRD,LADBWR)                                        -
     7                                                 ,(43,ORGCOR,LORGCO),-
     8           (44,PAGE,LPAGE),(45,DEBUG,LDEBUG),(46,BCT,LBCT),          -
     9           (47,ADB,LADB),(48,ENDADB,LENDAD)
     *DISK AREA OUPUT1(20),OUPUT2(20),OUPUT3(20),OUPUT4(20),OUPUT5(20),    -
     1          OUPUT6(20),INPUT(50)
     *IF(DEBUG.LT.1)GO TO 5
A     DISPLH "PUTBYT",0;
     *IF(DEBUG.LT.3)GO TO 5
A     DISPLH "PUTBYT:",2;
    5 *CONTINUE
C     FIRST WE ROTATE ADBOUT(ARRAY) LEFT 16 AND STICK IN OUBYT AND BUMP
C     BYTCNT(ARRAY) BY 1.
      LADBOU(ARRAY)=LADBOU(ARRAY).RTL.16
      LADBOU(ARRAY)=LADBOU(ARRAY).OR.LOUBYT
A     %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
      BYTCNT(ARRAY)=BYTCNT(ARRAY)+1
     *IF(DEBUG.LT.3)GO TO 15
A     DISPLH "PUTBYT1",2;
   15 *CONTINUE
     *IF (BYTCNT(ARRAY).LT.4)GO TO 100
```

```
C      ADB WORD IS FULL.
       OUPTW=OUPTWA(ARRAY)
C      AGAIN WE RUN INTO THE CFD RESTRICTION ON DIMENSIONING. EXCUSE THE
C      DELAY.
       LT6=LOUPTW.SHR.6
       LT5=OFF.TURN ON..LAST.6
A      %DUMMY ASK STATEMENT TO FORCE DEALLOCATION OF REGISTERS.
       LT5=LT5.AND.LOUPTW
       OUTBUF(T5+1,T6+1,ARRAY)=ADBOUT(ARRAY)
       BYTCNT(ARRAY)=0
       ADBOUT(ARRAY)=0
       T6=OUPTW+1
       OUPTWA(ARRAY)=T6
      *IF(OUPTW+1.LT.4096)GO TO 100
C      OUTBUF(ARRAY) IS FULL.
      *IF(DEBUG.LT.1)GO TO 25
A      DISPLH "COREFULL",0;
      *IF(DEBUG.LT.3)GO TO 25
A      DISPLH " ", 2;
   25 *CONTINUE
       T6=OUPAGE(ARRAY)
      *IF(ARRAY.EQ.1)WRITE(64,OUTBUF(1,1,1),OUPUT1(T6),3)
      *IF(ARRAY.EQ.2)WRITE(64,OUTBUF(1,1,2),OUPUT2(T6),3)
      *IF(ARRAY.EQ.3)WRITE(64,OUTBUF(1,1,3),OUPUT3(T6),3)
      *IF(ARRAY.EQ.4)WRITE(64,OUTBUF(1,1,4),OUPUT4(T6),3)
      *IF(ARRAY.EQ.5)WRITE(64,OUTBUF(1,1,5),OUPUT5(T6),3)
      *IF(ARRAY.EQ.6)WRITE(64,OUTBUF(1,1,6),OUPUT6(T6),3)
      *WAIT 64
       T6=OUPAGE(ARRAY)
       T6=T6+3
       OUPAGE(ARRAY)=T6
C      WE WROTE OUT EVERYTHING BUT THE LAST PAGE OF THE BUFFER. SAVE
C      THAT AT THE TOP OF THE BUFFER FOR FUTURE USE.
      *DO 40 T6=1,16
   40  OUTBUF(*,T6,ARRAY)=OUTBUF(*,T6+48,ARRAY)
       OUPTWA(ARRAY)=1024
C       THE REST OF THE BUFFER REALLY DOESNT HAVE TO BE ZEROED OUT,
C      BUT IT EXECUTES SUPER FAST AND MIGHT SAVE SOME CONFUSION
C      SOME DAY.
      *DO 50 T6=17,64
   50  OUTBUF(*,T6,ARRAY)=0
  100 *CONTINUE
      *IF(DEBUG.LT.1) GO TO 105
A      DISPLH "EPUTBYT:",0;
      *IF(DEBUG.LT.3)GO TO 105
A      DISPLH " ", 2;
  105 *CONTINUE
      *RETURN
      *END
      *SUBROUTINE RDPRM
      *PE INTEGER CNTRL(*,6),OUTBUF(*,64,6),PINTI(*),INBUF(*,128),          -
      1          TIME(*),OLDTIM(*),                                         -
      1                      SAVBCT,SAVPTW,OUPAGE(6),TSTEPS(6),SCANS,       -
      2            OUPTWA(6),      OTIMEA(6),ORGADB,INBUFI(8192)
      *CU INTEGER ADBBUF(8),ARRAY,INPTB,INPTW,SAVADB,ADBOUT(6),OUPTW,       -
      1            BYTS,WORDS,T1,T2,T3,T4,T5,T6,      IT,PRTIAL,ADDRS,      -
      2            WORD,      BYTCNT(6),ADBWRD,INBYT,OUBYT,ORGCOR,PAGE,     -
      3            DEBUG,BCT,ADB,ENDADB
      *CU LOGICAL LADBBU(8),LARRAY,LINPTB,LINPTW,LSAVAD,LADBOU(6),LOUPTW,-
      1            LBYTS,LWORDS,LT1,LT2,LT3,LT4,LT5,LT6,LOUBYT,LIT,LPRTIA,-
      2            LADDRS,LWORD,LINBYT,LBYTCN(6),LADBWR,LORGCO,LPAGE        -
      3            ,LDEBUG,LBCT,LADB,LENDAD
      *EXTERNAL GETBYT,PUTBYT,CNVTIM
      *COMMON/MAIN/CNTRL,OUTBUF,INBUF,PINTI,TIME,OLDTIM,SAVBCT,SAVPTW,     -
      1      TSTEPS,SCANS,OUPTWA,OUPAGE,OTIMEA,ORGADB
      *EQUIVALENCE(1,ADBBUF(1),LADBBU(1)),(9,ARRAY,LARRAY),                 -
      1            (10,INPTB,LINPTB),                                       -
      1            (!1,INPTW,LINPTW),(12,SAVADB,LSAVAD),                    -
      1            (13,ADBOUT(6),LADBOU(6))                                 -
      2            ,(19,OUPTW ,LOUPTW),(20,BYTS,LBYTS),(21,WORDS,LWORDS), -
      3            (22,T1,LT1),(23,T2,LT2),(24,T3,LT3),(25,T4,LT4),         -
      4            (26,T5,LT5),(27,T6,LT6),(28,OUBYT,LOUBYT),(29,IT,LIT), -
      5            (30,PRTIAL,LPRTIA),(31,ADDRS,LADDRS),(32,WORD,LWORD),    -
      6            (33,INBYT,LINBYT),(34,BYTCNT(1),LBYTCN(1)),              -
      6            (40,ADBWRD,LADBWR)                                       -
      7                                                 ,(43,ORGCOR,LORGCO),-
      8            (44,PAGE,LPAGE),(45,DEBUG,LDEBUG),(46,BCT,LBCT),         -
      9            (47,ADB,LADB),(48,ENDADB,LENDAD)
      *EQUIVALENCE (INBUF(1,1),INBUFI(1))
      *DISK AREA OUPUT1(20),OUPUT2(20),OUPUT3(20),OUPUT4(20),OUPUT5(20), -
      1          OUPUT6(20),INPUT(50)
       DEBUG=0
      *IF (DEBUG.LT.1) GO TO 10
A      DISPLH"RDPRM",0;
A      DISPLH "DEBUG",2;
   10 *CONTINUE
      *RETURN
      *END
      *SUBROUTINE REALE(IN,OUT)
      *PE REAL IN(*),OUT(*)
      *CU INTEGER DEBUG
      *EQUIVALENCE (3,DEBUG)
      *IF(DEBUG.LT.5)GO TO 10
A      DISPLH "REALE",0;
      *IF(DEBUG.LT.5)GO TO 10
A      LDL(0) $D49;
A      LDA IN(0);
A      DISPLH "IN",32;
   10 *CONTINUE
A      LDL(0) $D49;
A      LDA IN(0);
A      LDR $A;                    % SAVE IT.
A      RAB =0;                    % ELIMINATE THE SIGN SO WE CAN DO THE
A                                 % EXPONENT.
A      SHAR =56;                  % ISOLATE THE EXPONENT.
A      SBM =40:16;                % SUBTRACT THE 32 BIT OFFSET.
A      ADM =4000:16;              % ADD THE 64 BIT OFFSET.
A      SHAL =48;                  % PUT IT IN THE 64 BIT EXP. FIELD.
A      LDS $A;                    % SAVE IT.
A      LDA $R;                    % NOW FOR THE SIGN.
A      SHAR =63;
A      SHAL =63;                  % SIGN BIT IS ISOLATED.
A      OR $S;                     % NOW WE HAVE EXPONENT AND SIGN.
A      LDS $A;                    % SAVE IT.
A      LDA $R;                    % NOW FOR THE MANTISSA.
A      SHAL =40;                  % ISOLATE THE MANTISSA.
A      SHAR =16;                  % PUT IT IN 64 BIT EXP FIELD.
A      OR $S;                     % DONE.
A      LDL(0) $D50;
A      STA OUT(0);
      *IF(DEBUG.LT.5)GO TO 110
A      DISPLH "EREALE",0;
      *IF(DEBUG.LT.5)GO TO 110
A      LDL(0) $D50;
A      LDA OUT(0);
A      DISPLH "OUT",32;
  110 *CONTINUE
      *RETURN
      *END
      *FUNCTION ROWSUM(R)
      *PE REAL R,ROWSUM
      *CU INTEGER DEBUG
      *EQUIVALENCE (45,DEBUG)
      *CONTINUE
A      LDL(0) $D49;
A      LDA R(0);
A      LIT(0) 1,6,1;
A      LIT(1) =1;
A      ROWSUMLOOP;
A      RTL $A,0(1);
A      ADRN $R;
A      CSHL(1) 1;
A      TXEFM(0) ,ROWSUMLOOP;
A      STA RFUNVAL;
      *IF(DEBUG.LT.1)GO TO 20
A      DISPLH "EROWSUM",0;
      *IF(DEBUG.LT.2)GO TO 20
A      DISPLH "ROWSUM",16,RFUNVAL,RFUNVAL+63;
A      LDL(0) $D49;
A      LDA R(0);
A      DISPLH "R",32;
   20 *CONTINUE
      *RETURN
      *END
      *SUBROUTINE RUNFFT
C      THIS SUBROUTINE CALLS THE UNIVERSITY OF ILLINOIS (JIM STEVENS)
C      FFT ROUTINE AFTER SETTING UP THE PROPPER PARAMETERS. THE DATA
C      TO BE FFT'ED IS IN BUFF2. IT STARTS AT EITHER BUFF2+0 OR BUFF2
C      +35200 DEPENDING ON THE VALUE OF "NEW". THE NUMBER OF TIMEWINDOWS
C      IS "NGDCH" AND THE  TIME WINDOW SIZE IS IN "TWSZ".
      *PE INTEGER NBUFF1(*,64),FINSCN(*),COMP(*),TOTSCN(*),PINT1(*),    -
      1           PINT2(*),TIME(*),OTIME(*),TWTIME(*),PEN(*),CNTRL(*,6)
      *PE REAL SAVE(*)
      *PE REAL GLCHFT(*),VARFT(*),BUFF2(*,550,2),BUFF3(*,640),PREAL1(*), -
      1        PREAL2(*),ALLMSQ(*),TVARFT(*)
      *PE INTEGER LOFREQ,HIFREQ,IBUFF1(4096),IBUFF3(*,640),ABUFF2(70400),-
      1           CHGOOD(80),SITEGD(80),SITES(80)
      *PE REAL CHMSQ(80),RBUFF1(4096),ROWSUM,RBUFF2(70400)
      *CU REAL RADBBU(8)
      *CU INTEGER ADBBUF(8),COREPT,      BYTE,ADBWRD,ARRAY,DEBUG,TWSZ,   -
      1           OVLAP,NCHAN,NSITE,NROWS,DIFFR,DIFFW,NEW,OLD,GAP,TSCANS,-
      2          INDEX1,INDEX2,INDEX3,INDEX4,T1,T2,T3,T4,T5,T6,CH,       -
      3          OFFSET,INBYT,NGDCH,TWSZR,NGDST,NGDR,F,BF3PE,NGT,PAGE,T7
      *CU LOGICAL LADBBU(8),LCOREP,LAST16,LBYTE,LADBWR,LARRAY,LDEBUG,    -
      1           LTWSZ,LOVLAP,LNCHAN,LNSITE,LNROWS,LDIFFR,LDIFFW,LNEW,  -
      2           LOLD,LGAP,LTSCAN ,LT1,LT2,LT3,LT4,LT5,LT6,LCH,LOFFSE,  -
      3           LINBYT,LF,LNGDCH,LTWSZR,LNGDST,LNGDR,LNGT,LT7
      *EXTERNAL GTDATA,C16T64,C64T32,ROWSUM,C32T64
      *COMMON/MAIN2/NBUFF1,BUFF2,BUFF3,FINSCN,COMP,TOTSCN,PINT1,PINT2,   -
      1      TIME,OTIME,TWTIME,PEN,CNTRL,GLCHFT,VARFT,PREAL1,PREAL2,     -
      2      ALLMSQ,TVARFT,LOFREQ,HIFREQ,SITEGD,SITES,CHMSQ
      *EQUIVALENCE (NBUFF1(1,1),RBUFF1(1),IBUFF1(1)),(BUFF2(1,1,1),      -
      1             ABUFF2(1),RBUFF2(1)),(CHGOOD(1),SITEGD(1)),          -
      2           (BUFF3(1,1),IBUFF3(1,1))
      *EQUIVALENCE(1,RADBBU(1))
      *EQUIVALENCE (1,ADBBUF(1),LADBBU(1)),(9,COREPT,LCOREP),(10,BYTE,   -
      1           LBYTE),(11,ADBWRD,LADBWR),(12,ARRAY,LARRAY),(13,TWSZ,  -
      2           LTWSZ),(14,OVLAP,LOVLAP),(15,NCHAN,LNCHAN),(16,NSITE,  -
      3           LNSITE),(17,NROWS,LNROWS),(18,DIFFR,LDIFFR),(19,DIFFW, -
      4           LDIFFW),(20,NEW,LNEW);(21,OLD,LOLD),(22,GAP,LGAP),(23, -
      5           TSCANS,LTSCAN),(24,INDEX1),(25,INDEX2),(26,INDEX3),(27,-
      6           INDEX4),(28,T1,LT1),(29,T2,LT2),(30,T3,LT3),(31,T4,    -
      7           LT4),(32,T5,LT5),(33,T6,LT6),(34,CH,LCH),(35,OFFSET,   -
      8           LOFFSE),(36,INBYT,LINBYT),(37,F,LF),(38,NGDCH,LNGDCH), -
      9           (39,TWSZR,LTWSZR),(40,NGDST,LNGDST),(41,NGDR,LNGDR),   -
      0           (42,BF3PE),(43,NGT,LNGT),(44,LAST16),(45,DEBUG,LDEBUG),-
      1           (46,PAGE),(47,T7,LT7)
      *DISK AREA INDM2(20),OUTDM2(40),CONPRM(1)
      *IF(DEBUG.LT.1)GO TO 10
A      DISPLH "RUNFFT",0;
      *IF(DEBUG.LT.2)GO TO 10
A      DISPLH ,2;
A      DISPLH "ERUNFFT",0;
   10 *CONTINUE
      *TRANSFER(8) SAVE(1)=RADBBU(1)
```

```
C      SAVE THE ADB LOCATIONS DESTROYED BY FFT ROUTINE.
A      SLIT(0) SIZE;
A      LDL(1) TWSZ;
A      STORE(0) $C1;
A      SLIT(0) NUMBER;
A      LDL(1) NGDCH;
A      STORE(0) $C1;
A      SLIT(0) ARGLIST+2;
A      SLIT(1) BUFF2;
A      LDL(2) OLD;
A      ALIT(2) -1;
A      ZERXT(2) ,1;
A      ALIT(1) 35200;
A      STORE(0) $C1;
A      EXTERNAL FFT;
      *IF(DEBUG.LT.2)GO TO 20
A      SLIT(1) ARGLIST+2;
A      LOAD(1) $C1;
A      CSHR(1) 6;
A      LDA 0(1);
A      SLIT(3) ARGLIST;
A      LOAD(3) $C0;
A      LOAD(0) $C0;
A      ALIT(3) 1;
A      LOAD(3) $C1;
A      LOAD(1) $C1;
A      ALIT(3) 1;
A      LOAD(3) $C2;
A      DISPLH "B4 FFT:",33;
   20 *CONTINUE
A      CLC(3);
A      SLIT(3) FFT;
A      SLIT(2) ARGLIST;
A      EXCHL(3) $ICR;
A      CACRB 10;
      *IF(DEBUG.LT.2)GO TO 30
A      SLIT(1) ARGLIST+2;
A      LOAD(1) $C1;
A      CSHR(1) 6;
A      LDA 0(1);
A      DISPLH "AFTER:",33;
   30 *CONTINUE
A      SKIP ,EARGLIST;
A      ARGLIST:
A      DATA SIZE,NUMBER,0;
A      SIZE:WDS 1;
A      NUMBER:WDS 1;
A      EARGLIST:;
      *TRANSFER(8) WADBBU(1)=SAVE(1)
  110 *CONTINUE
      *RETURN
      *END
```